KOREAN PATENT ABSTRACT (KR)

PUBLICATION

(51) IPC Code: H04L 12/28
(11) Publication No.: P1999-0064264 (43) Publication Date: 26 July 1999
(21) Application No.: 10-1998-0702754 (22) Application Date: 15 April 1998
(86) International Application No.: PCT/GB1996/02317 (20 September 1996)
(87) International Publication No.: WO 1997/18635 (22 May 1997)

(71) Applicant:
International Business Machines Corporation;
Armonk, NY 10504 US.

(72) Inventor:
Peercy Michael; US.
Primm Michael; US.
Reed Benjamin; US.
Welch Steven; US.
Medford Mitchell; US.

(54) Title of the Invention:

Information Handling System for Allowing a Generic Web Browser to Access Servers of a Plurality of Different Protocol Types

Abstract:

An information handling system enables a client computer device to access information located at a server computer device over a network between the client and server computer devices. The client computer device accesses the server computer device using HyperText Transfer Protocol HTTP and presents information received from the server computer device to a user of the client computer device using HyperText Markup Language software. The information handling system includes the network between the client and server computer devices, the network using a network transmission protocol other than HTTP and using a network data format other than HTML, and a transmission protocol converter which converts HTML/HTTP information sent out from or received by at least one of the client computer device and the server computer device, via the network, to the network transmission protocol and the network data format.

(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(51) Int. Cl.[6]
H04L 12/28

(11) 공개번호 특1999-0064264
(43) 공개일자 1999년07월26일

(21) 출원번호 10-1998-0702754
(22) 출원일자 1998년04월15일
번역문제출일자 1998년04월15일
(86) 국제출원번호 PCT/GB1996/02317
(87) 국제공개번호 WO 1997/18635
(86) 국제출원출원일자 1996년09월20일
(87) 국제공개일자 1997년05월22일
(81) 지정국 EP 유럽특허 : 오스트리아 벨기에 스위스 독일 덴마크 스페인 핀랜드 프랑스 영국 그리스 이탈리아 룩셈부르크 모나코 네덜란드 포르투칼

국내특허 : 아일랜드 브라질 캐나다 중국 체크 헝가리 일본 대한민국 폴란드

(30) 우선권주장 8/558,626 1995년11월14일 미국(US)
8/558,627 1995년11월14일 미국(US)
8/558,628 1995년11월14일 미국(US)
8/558,631 1995년11월14일 미국(US)
~~8/558,627 1995년11월14일 미국(US)~~
~~8/558,628 1995년11월14일 미국(US)~~
~~8/558,631 1995년11월14일 미국(US)~~

(71) 출원인 인터내셔널 비지네스 머신즈 코포레이션
미국 10504 뉴욕주 아몬크
(72) 발명자 피어시 마이클
미국 캘리포니아주 95014 쿠페르티노 #에이28 노쓰 풋힐 블러바드 10330
프링 마이클
미국 노쓰캐롤라이나주 27502 아펙스 브루크 크리크드라이브 409
리드 벤자민
미국 캘리포니아주 95064 산타 크루즈 코쉬랜드 웨이 507
웰치 스티븐
미국 캘리포니아주 95020 질로이 캐나다 로드 3330
메드포드 미첼
미국 노쓰캐롤라이나주 27511 캐리 스트래쓰버그 레인 109
(74) 대리인 김창세, 장성구

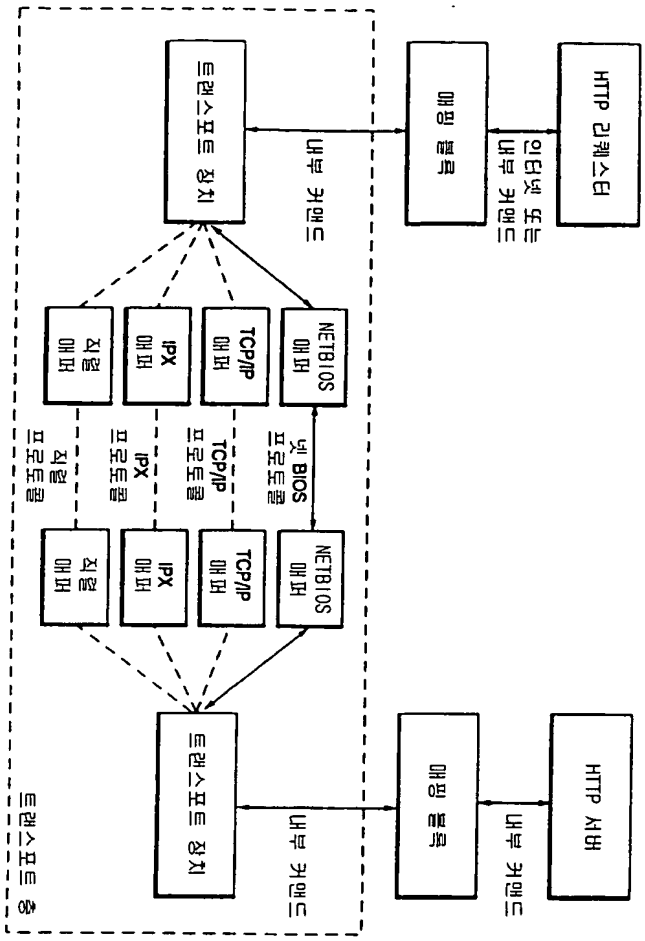
심사청구 : 있음

(54) 일반 웹 브라우저가 다수의 상이한 프로토콜 유형들을 갖는 서버를 액세스할 수 있게 하는 정보 운용 시스템

요약

본 발명의 정보 운용 시스템은 클라이언트 컴퓨터 장치와 서버 컴퓨터 장치 사이에서 네트워크를 통해 클라이언트 컴퓨터 장치가 서버 컴퓨터 장치에 위치된 정보를 액세스할 수 있게 한다. 클라이언트 컴퓨터 장치는 하이퍼텍스트 전송 프로토콜(HTTP)을 사용하여 서버 컴퓨터 장치를 액세스하고, 서버 컴퓨터 장치로부터 수신된 정보를 하이퍼텍스트 마크업 언어 소프트웨어를 사용하여 서버 컴퓨터 장치로부터 클라이언트 컴퓨터 장치의 사용자에게 제공한다. 본 정보 운용 시스템은 클라이언트 컴퓨터 장치와 서버 컴퓨터 장치간에 네트워크를 포함하는데, 이 네트워크는 HTTP 이외의 네트워크 전송 프로토콜을 사용하고, HTML 이외의 네트워크 데이터 포맷을 사용하며, 네트워크를 통해 클라이언트 컴퓨터 장치 및 서버 컴퓨터 장치중 적어도 하나로부터 송신되거나 그에 의해 수신된 HTML/HTTP 정보를 네트워크 전송 프로토콜 및 네트워크 데이터 포맷으로 변환하는 전송 프로토콜 변환기를 이용한다.

대표도



명세서

기술분야

본 발명은 다수의 망연결된 정보 처리 시스템(information processing systems)을 갖는 정보 운용 시스템(information handling systems)에 관한 것으로, 보다 구체적으로는, 클래스 라이브러리(class libraries)를 사용하여 월드 와이드 웹(World Wide Web)으로 알려진 것에 대한 서버를 개발하기 위한 객체 지향형 기술(object oriented technology)에 관한 것이다.

배경기술

과거 20년 동안에, 인터넷(Internet)으로 알려진 전역적으로 연결된 컴퓨터 네트워크, 특히, 이 인터넷의 상부에 제공되는 설비들중 하나인 월드 와이드 웹(World Wide Web;WWW)의 사용이 폭발적으로 성장하였다. WWW는 다수의 정보 페이지들 또는 파일들을 포함하며, 이 정보는 다수의 상이한 서버 컴퓨터 시스템들에 대해 배포된다. 이들 페이지상에 저장된 정보는, 예를 들면, 회사의 조직 세부 내용, 접촉 데이터, 제품 데이터 및 회사 소식들일 수 있다. 이 정보는 텍스트, 그래픽 오디오 데이터 및 비디오 데이터의 조합으로 사용자의 컴퓨터 시스템('클라이언트 컴퓨터 시스템')에 제공될 수 있다. 각 페이지는 URL(a Uniform Resource Locator)에 의해 식별된다. 이 URL은 서버 머신(server machine)과 해당 머신에 대한 특정 파일 또는 페이지를 표시한다. 단일의 서버상에는 다수의 페이지 또는 URL이 상주할 수 있다.

WWW를 사용하려면, 클라이언트 컴퓨터 시스템은 그래픽 웹 브라우저(a graphical Web browser)로서 알려진, 웹익스플로러(WebExplorer)(IBM사로부터 OS/2 오퍼레이팅 시스템의 일부로서 제공됨) 또는 넷스케이프 커뮤니케이션즈사로부터 구할 수 있는 내비게이터(Navigator) 프로그램과 같은 한편의 소프트웨어를 실행시켜야 한다. '웹익스플로러', 'OS/2' 또는 'IBM'이라는 용어는 IBM사의 등록상표이고, '내비게이터' 및 '넷스케이프'는 넷스케이프 커뮤니케이션즈사의 등록상표이다. 클라이언트 컴퓨터 시스템은 브라우저와 상호작용하여 특정의 URL을 선택하며, 이것은 또한 브라우저로 하여금 그 URL 또는 페이지에 대한 요구를 이 URL내에 식별된 서버로 송신하게 한다. 통상 서버는, 요구된 페이지를 검색(retrieve)하여 그 페이지에 대한 데이터를 요구 클라이언트 컴퓨터 시스템으로 되전송함으로써 요구에 응답한다(클라이언트/서버 상호작용은 하이퍼텍스트 전송 프로토콜(HyperText Transport Protocol;http)에 따라 수행된다). 그리고 나서, 이 페이지는 클라이언트 스크린상에서 사용자에게 디스플레이된다. 클라이언

트는 또한 서버로 하여금 애플리케이션을 시작하게 하거나, 예를 들면, 특정 주제에 관련되는 WWW 페이지를 탐색하게 할 수도 있다.

대부분의 WWW 페이지는 하이퍼텍스트 마크업 언어(HyperText Mark-up Language;HTML)로 알려진 언어로 작성된 컴퓨터 프로그램에 따라 포맷된다. 이 프로그램은 클라이언트의 그래픽 브라우저에 의해 디스플레이될 데이터와 브라우저에게 데이터를 디스플레이하는 방법을 지시하는 포매팅 커맨드를 포함한다. 따라서, 통상의 웹 페이지는 태그(tag)로 알려지는 내장 포매팅 커맨드(embedded formatting commands)와 함께 텍스트를 포함하며, 이것은 폰트 사이즈(font size), 폰트 스타일(예를 들면, 이태리체인지 또는 진하게(bold)인지), 텍스트를 배치하는 방법 등을 제어하는데 사용될 수 있다. 웹 브라우저는 텍스트를 지정된 포맷에 따라 디스플레이하기 위해 HTML 스크립트를 '파싱(parsing)'한다. HTML 태그는 또한 그래픽, 오디오 및 비디오가 클라이언트의 브라우저에 의해 사용자에게 나타내어지는 방식을 표시하는데 사용된다.

대부분의 웹 페이지는 또한 다른 웹 페이지들에 대한 하나 이상의 조회들(one or more reference)을 포함하며, 이것은 원래의 페이지와 동일한 서버상에 있을 필요는 없다. 통상 이들 조회는 사용자가 스크린상의 특정 위치를 선택함으로써, 전형적으로, 마우스 제어 버튼을 (더블) 클릭함으로써 활성화된다. 이들 조회 또는 위치지정은 하이퍼링크(hyperlinks)로 알려져 있으며, 통상, 특정 방식(예를 들면, 하이퍼링크와 연관된 모든 텍스트를 상이한 컬러로 할 수 있음)으로 브라우저에 의해 표시된다. 사용자가 하이퍼링크를 선택하면, 이 때 조회된 페이지가 검색되어 현재 디스플레이된 페이지를 대체한다.

HTML 및 WWW에 대한 보다 상세한 정보는, 1994년 12월, 닥터 돕스 저널, 18-26페이지, 더글라스 맥아더의 '월드 와이드 웹과 HTML'에서, 그리고 아이안 그래함의 'HTML 소스북'(1995년, 뉴욕, 존 윌리)에서 찾아볼 수 있다.

HTML을 디스플레이하고 수신하는 일반 웹 브라우저는 애플리케이션 개발자들에게 놀라운 능력을 제공한다. 사용자 인터페이스를 구현하는데 브라우저를 사용하면, 두 개의 큰 장점이 있다. 즉, 통신 프로토콜(HTTP)이 구축되고, 사용자 인터페이스 환경(HTML)이 구축된다. 통신수단 및 사용자 인터페이스는 모두 매우 비싼 애플리케이션 개발 영역인데, 그 주된 이유는 통신수단 및 사용자 인터페이스가 극히 플랫폼에 종속적이기 때문이다. 다수의 애플리케이션들은 이 두 문제점으로 인해 그들을 사용할 수 있는 모든 고객에게 이르지 못한다.

현재 HTTP TCP/IP 이외의 어떤 통신 프로토콜을 사용하여 전송하는 웹 브라우저/서버 시스템은 없다. 그러나 다수의 컴퓨터 설비들은 넷바이오스(NetBIOS), IPX, 및 직렬 라인들과 같은 다양한 통신 프로토콜을 이용한다. 이들 설비의 사용자는 TCP/IP 접속을 사용할 필요없이 웹 브라우저/서버 기술을 사용하기를 바랄 수도 있다. 이러한 기술은 그 사양에 의해 제한된다.

이러한 기술은 특히 시스템 관리 환경에서 중요하다. 효율적인 시스템 관리는 크고 작은 컴퓨터 설비들의 동작에 중요하다. 시스템 관리 태스크를 개선하기 위해 다수의 애플리케이션들이 개발되었다. 이들은, 시스템 사용자 또는 관리자에게 그들의 관리에 있어 더욱 많은 능력을 제공하기 위해 컴퓨터 시스템의 비밀 하드웨어(및 오퍼레이팅 시스템)-종속 관측 및 제어를 추출해서 합치려고 시도한다.

오늘날의 시스템 관리 소프트웨어에서는 관측 및 제어가 그래픽 사용자 인터페이스(graphical user interfaces;GUIs)를 통해 디스플레이되고 제어된다. 본 발명과 관련하여, 이들 시스템 관리 애플리케이션은 다음과 같은 두 가지 문제점을 갖는다. 즉, (1) GUI는 지정된 시스템상에서 실행되고 제한된 프로토콜을 통해 기본 시스템 관리 객체들과 통신하도록 제한되며, (2) 시스템 관리 애플리케이션을 다른 시스템 또는 프로토콜로 변경하려면 그 새로운 플랫폼에 대해 GUI를 재코딩(recoding)할 필요가 있다. 현재 기술은 단지 지원하는 시스템에 시스템 관리 데이터를 제공할 뿐이다. 따라서, 현재 기술에서는 강력하고 유연성있는 시스템 관리 인터페이스에 대한 장벽이 있다.

<u>발명의 개요</u>

본 발명은 클라이언트 컴퓨터 장치(a client computer device)가 서버 컴퓨터 장치(a server computer device)에 위치된 정보를 액세스할 수 있게 하는 정보 운용 시스템(an information handling system)을 제공하되, 상기 클라이언트 컴퓨터 장치는 하이퍼텍스트 전송 프로토콜(HyperText Transfer Protocol;HTTP)을 사용하여 상기 서버 컴퓨터 장치와 통신하는 수단과, 하이퍼텍스트 마크업 언어(HyperText Markup Language;HTML)를 사용하여 클라이언트 컴퓨터 장치 사용자에게 상기 서버 컴퓨터 장치로부터 수신된 정보를 제공하는 수단을 포함하고, 상기 정보 운용 시스템은 상기 클라이언트 컴퓨터 장치와 상기 서버 컴퓨터 장치간의 네트워크(a network)-상기 네트워크는 HTTP 이외의 네트워크 전송 프로토콜을 사용하고, HTML 이외의 네트워크 데이터 포맷을 사용함-와, 상기 네트워크를 통해서, 상기 클라이언트 컴퓨터 장치 및 상기 서버 컴퓨터 장치로부터 송신되온 또는 그에 의해 수신된 HTML/HTTP 정보를 상기 네트워크 전송 프로토콜 및 상기 네트워크 데이터 포맷으로 변환하는 전송 프로토콜 변환 수단(a transmission protocol conversion means)을 포함한다.

바람직하기로는, 본 발명은 또한 상기한 시스템을 포함하는데, 상기 클라이언트 컴퓨터 장치의 정보를 제공하는 수단은 특정 유형의 그래픽 사용자 인터페이스를 사용하며, 상기 클라이언트 컴퓨터 장치는 인터넷에 접속되고, 상기 네트워크는 인터넷과 서버 컴퓨터 장치 사이에 위치되며, 상기 전송 프로토콜 변환 수단은 객체 지향형 컴퓨터 소프트웨어를 실행하고, 상기 객체 지향형 컴퓨터 소프트웨어는 상기 클라이언트 컴퓨터 장치로부터의 HTML 요구 데이터를 수신하여 이 수신된 HTML 요구 데이터를 네트워크 데이터 포맷으로 변환하는 HTML 객체와, 상기 클라이언트 컴퓨터 장치와 서버 컴퓨터 장치간에 인터넷 통신을 개시하여 유지하고, 클라이언트 컴퓨터 장치로부터 수신된 HTML 요구 데이터를 파싱하며, 상기 클라이언트 컴퓨터 장치에 의해 사용된 특정 그래픽 사용자 인터페이스의 신원(identity;ID)을 검출하고, 요구된 데이터 및 검출된 그래픽 사용자 인터페이스 ID를 HTML 객체로 전송하며, 상기 HTML 객체와 상기 클라이언트 컴퓨터 장치 사이에 상기 네트워크 및 인터넷을 통해 HTML 데이터를 전송하는 HTTP 객체를 포함한다.

두 번째 측면에서, 본 발명은 정보 운용 시스템용 프로그램 요소(a program element)를 제공하는데, 상

기 정보 운용 시스템상에서 실행되었을 때, (a) HTML 및 HTTP와 (b) 이 HTML 및 HTTP 이외의 커맨드 및 프로토콜에 대해 데이터를 매핑하는 매핑 블록(a mapping block)을 포함하며, 상기 매핑 블록은 HTML 객체 및 HTTP 객체를 가지며, 상기 HTML 객체는, URL 및 다른 요구 데이터를 수신하고, 수신된 요구를 HTML 및 HTTP 이외의 내부 커맨드 및 프로토콜로 형성하며, HTML 및 HTTP 이외의 내부 커맨드 및 프로토콜로 응답(reply)을 수신하고, 수신된 응답을 HTML로 변환하며, 이러한 HTML을 상기 HTTP 객체로 송신하고, 상기 HTTP 객체는, 인터넷 통신을 개시 및 유지하며, 인터넷을 통해 HTTP 요구를 수신하고, 수신된 요구를 형성하는 URL 및 연관된 데이터를 파싱하며, 이 URL 및 연관 데이터와 브라우저 프로그램 ID를 상기 HTML 객체로 전송하고, 상기 HTML 객체로부터 HTML 소스를 수신하며, HTML 소스를 HTTP 응답들로서 인터넷으로 복귀한다.

본 발명은 또한 인터넷상의 어디서 접속된 어떤 웹 브라우저라도 시스템 관리 프로토콜 이벤트들(events)을 브라우즈 및 제어하기 위한 그래픽 사용자 인터페이스로서 기능할 수 있도록 한다.

도면의 간단한 설명

본 발명의 목적들중 몇몇이 설명되었으며, 나머지는 이하와 같은 첨부 도면과 관련하여 상세한 설명을 진행함에 따라 명백해질 것이다.

도 1a는 전형적인 퍼스널 컴퓨터 시스템의 사시도,

도 1b는 본 발명에 따른 장치 및 방법을 구체화한 정보 운용 네트워크를 도시한 도면,

도 2 및 2a 내지 2d는 도 1a에 도시된 컴퓨터 시스템에 대한 일체화된 플래너 보드의 개략적인 블록도,

도 3 및 3a 내지 3c는 도 1a에 도시된 컴퓨터 시스템에 대한 다른 플래너 보드의 개략적인 블록도,

도 4, 4a 및 4b는 도 3에 도시된 다른 플래너 보드와 함께 사용하기 위한 프로세서 카드의 개략적인 블록도,

도 5는 본 발명에 따라 도 1b에 도시된 정보 운용 네트워크를 통해 시스템 구성 정보를 전달하기 위해 수행되는 단계들을 도시한 흐름도,

도 6은 도 5에 도시된 흐름도에 따라 수행되는 단계들중 몇몇을 보다 상세히 도시한 흐름도,

도 7은 예시적인 프레임워크 메카니즘의 카테고리를 도시한 도면,

도 8 내지 12는 도 7에 도시된 예시적인 프레임워크 메카니즘의 클래스를 도시한 도면,

도 13은 도 7 내지 12에 도시된 예시적인 프레임워크 메카니즘에 대한 객체를 도시한 도면,

도 14a 및 14b는 도 1b에 도시된 네트워크를 통해 실행되는 본 발명에 따른 시스템 관리의 실시예를 도시한 도면,

도 15는 도 14a 및 14b에 도시된 매핑 블록 요소를 도시한 도면이다.

실시예

이하에서는 본 발명의 배경에 관련되는 기술에 대한 설명이 계속된다. 충분한 지성과 지식을 갖춘 자는 이러한 기술 배경에 대한 정보를 훑어보거나 읽지 않을 수도 있으나, 이 정보는 본 발명을 이해하는 데 중요하므로, 검토해볼 것을 권장한다.

이하에서는 본 발명의 바람직한 실시예들이 도시된 첨부 도면을 참조하여 본 발명이 보다 충분히 설명되는데, 후속되는 설명을 시작함에 있어, 당 분야에 숙련된 자라면 본 발명의 바람직한 결과들을 여전히 성취하면서 본 명세서에 기술된 본 발명을 수정할 수도 있음이 이해되어야 한다. 따라서, 이하의 설명은 당분야에 숙련된 자에게 제공되는 대강의 교시물로서 이해되어야지 본 발명을 제한하는 것으로서 이해되어서는 안된다.

하드웨어 환경

일반적인 퍼스널 컴퓨터 시스템, 특히 IBM 퍼스널 컴퓨터는 현대 사회의 많은 부분에 컴퓨터 능력을 제공하기 위해 광범위하게 사용되고 있다. 퍼스널 컴퓨터 시스템은 통상, 시스템 프로세서를 갖는 시스템 유닛, 디스플레이 모니터, 키보드, 하나 이상의 디스켓 드라이브, 고정 디스크 저장장치, '마우스'와 같은 선택사양적인 포인팅 장치 및 선택사양적인 프린터를 포함하는 데스크 탑 컴퓨터, 플로어 스탠딩(floor standing) 컴퓨터 또는 포터블 컴퓨터로서 정의될 수 있다. 이들 시스템은 주로 단일의 사용자 또는 소그룹의 사용자들에게 독립적인 컴퓨팅 능력을 제공하도록 설계되며, 개인 또는 사업체들에 의해 구매가능하도록 저렴한 가격으로 되어 있다. 이러한 퍼스널 컴퓨터 시스템의 예들은 IBM사의 퍼스널 컴퓨터(PERSONAL COMPUTER), 퍼스널 컴퓨터 XT(PERSONAL COMPUTER XT), 퍼스널 컴퓨터 AT(PERSONAL COMPUTER AT) 및 IBM사의 퍼스널 시스템/2(PERSONAL SYSTEM/2)(이후 각각 IBM PC, XT, AT 및 PS/2로 지칭함) 모델 25, 30, 50, 55, 57, 60, 65, 70, 80, 90 및 95라는 등록상표명으로 판매된다.

이들 시스템은 두 개의 일반적인 패밀리들로 분류될 수 있다. 제 1 패밀리는 통상 패밀리 1 모델(Family 1 Models)이라 지칭되며, AT 컴퓨터 및 다른 'IBM 호환' 기기들로 예시되는 버스 아키텍처를 사용한다. 제 2 패밀리는 패밀리 2 모델이라 지칭하며, IBM사의 PS/2 모델 50 내지 95로 예시되는 IBM사의 마이크로채널(MICRO CHANNEL) 버스 아키텍처를 사용한다. 패밀리 1 및 패밀리 2에서 사용된 버스 아키텍처는 당 분야에 잘 알려져 있다.

IBM PC 및 XT는 IBM 퍼스널 컴퓨터 계열의 제 1 모델들이었고 인텔 8088 프로세서를 사용하였다. IBM 퍼스널 컴퓨터 시스템에 대한 다음의 현저한 변화는 인텔 80286 프로세서를 사용한 AT였다. PS/2 계열은 인텔 프로세서들중 몇 개에 걸쳐져 있다. IBM PC 및 XT와 유사한 시스템은 PS/2 모델 30의 버전이며

인텔 8086 프로세서를 사용하였다. PS/2 모델 50 및 60은 모두 인텔 80286 프로세서를 사용하였다. IBM PS/2 모델 80과 IBM PS/2 모델 70의 특정 버전에서는 인텔 80386 프로세서가 사용되었다. IBM PS/2 모델 70의 나머지 버전 및 PS/2 모델 90 XP 486 및 95 XP 486은 인텔 80486 프로세서를 사용한다. 이들 모든 시스템에 있어서의 공통점들중 하나는 인텔 x86 계열 프로세서를 사용한다는 것이다. 통상 이용가능하고 잘 알려진 여러 가지의 소프트웨어 오퍼레이팅 시스템(예를 들면, DOS 또는 OS/2 오퍼레이팅 시스템)은 다양한 인텔 x86 계열 프로세서들상에서 동작할 수 있다.

또한, IBM PC와 같은 패밀리 1 모델의 최초의 퍼스널 컴퓨터 시스템을 시작함에 있어, 소프트웨어와 하드웨어간에 호환성을 성취하는 것이 가장 중요한 목적인 것으로 알려져 있었다. 이러한 목적을 달성하기 위해, 시스템 상주 코드(또한 '마이크로코드(microcode)'라고도 함)의 분리층(an insulation layer)이 하드웨어와 소프트웨어간에 확립되었다. 이러한 코드는 사용자의 애플리케이션 프로그램/오퍼레이팅 시스템과 하드웨어 장치간에 동작적인 인터페이스를 제공하여 하드웨어 장치들의 특징들에 대해 염려하는 사용자들을 안도케하였다. 결국, 코드는 BIOS로 발전하였으며, 새로운 하드웨어 장치들이 시스템에 부가될 수 있게 하면서, 하드웨어 장치들의 특성들로부터 애플리케이션 프로그램/오퍼레이팅 시스템을 분리시켰다. BIOS는 장치 드라이버가 특정 하드웨어 장치의 특성들에 종속적이지 않게 하면서 장치 드라이버에 하드웨어 장치에 대한 중간적인 인터페이스를 제공하기 때문에 BIOS의 중요성은 즉각적으로 증명되었다. BIOS는 컴퓨터 시스템과 일체화된 부분이고 시스템 프로세서 안팎으로 데이터의 이동을 제어하기 때문에 시스템 유닛의 시스템 플래너 보드상에 상주하며, 판독-전용 메모리(ROM) 또는 소거 및 프로그램 가능 판독-전용 메모리(EPROM)로 사용자에게 전달되었다. 예를 들어, 최초의 IBM PC에서 BIOS는 플래너 보드상에 상주하는 8Kbytes의 ROM(1킬로 바이트 또는 '1 Kbyte'는 1024 바이트를 말함)을 점유하였다. ROM에 부가하여, 플래너 보드는 시스템 프로세서, 주 랜덤 액세스 메모리(RAM), 및 이 보드상에 사실상 공면 관계(coplanar relationship)로 고정된 다른 구성요소들을 포함하였다. ROM은 또한 컴퓨터 시스템을 테스트하여 초기화하는데 사용되는 파워-온 셀프 테스트(a power-on self test;POST) 프로그램을 포함하였다. 컴퓨터 시스템 ROM에 상주하는 코드 전체는 '시스템 펌웨어(system firmware)' 또는 간단히 '펌웨어'로서 알려지게 되었다. 따라서, 펌웨어는 POST 부분과 BIOS 부분을 포함하였다. 때때로, BIOS는 POST 프로그램을 포함하도록 정의되었다.

퍼스널 컴퓨터 패밀리의 신 모델들이 도입됨에 따라, 펌웨어는 입/출력(I/O) 장치들과 같은 새로운 하드웨어 장치들을 지원하도록 갱신 및 확장되어야 했다. 예상할 수 있는 바와 같이, 펌웨어는 메모리 사이즈를 증가시키기 시작했다. 예를 들어, IBM 퍼스널 컴퓨터 AT의 도입으로, 펌웨어는 32Kbytes의 ROM을 필요로 하게 되었다. 마이크로채널 아키텍처를 갖는 IBM 퍼스널 시스템/2 컴퓨터 시스템의 도입으로, 상당히 새로운 BIOS, 즉, 고급 BIOS(Advanced BIOS;ABIOS)가 개발되었다. 그러나, 소프트웨어 호환성을 유지하기 위해 패밀리 1 모델로부터의 BIOS가 패밀리 2 모델에 포함되어야 했다. 패밀리 1 BIOS는 호환성(Compatibility) BIOS 또는 CBIOS로서 알려지게 되었다. 따라서, BIOS는 호환성 기본 입/출력 시스템(CBIOS) 및 고급 기본 입/출력 시스템(ABIOS)과 같은 하나를 초과하는 BIOS 유형을 포함하도록 진화되었다. 퍼스널 컴퓨터 시스템에 대한 현재의 구조적 정의들은 펌웨어에 대한 시스템 어드레스 공간(시스템 펌웨어 어드레스 공간)으로 최고 128Kbytes를 허용한다.

오늘날, 계속되는 신기술 개발에 의해, 퍼스널 컴퓨터 시스템은 더욱 복잡해지고 더욱 자주 개선되고 있다. 기술은 급속히 변화하고 새로운 I/O 장치들이 퍼스널 컴퓨터 시스템에 부가되고 있기 때문에 펌웨어 대한 수정 및 확장을 가하는 것이 퍼스널 컴퓨터 시스템의 성장 주기에 있어 중요한 문제가 되었다.

마이크로 채널 아키텍처의 도입으로, IBM은 프로그램가능 사양 선택(Programmable Option Select;POS)으로서 알려진 새로운 구성의 프로시쥬어를 제공하였다. POS는, DIP 스위치, 점퍼(jumpers) 및 헤더(headers)를 사용하여 시스템을 구성할 필요를 없앰으로써 시스템 성능 증대 설비 및 확장을 이전의 PC들에서보다 훨씬 더 용이하게 하고 훨씬 덜 혼란스럽게 하도록 설계된다. 저 파워를 사용하여, 배터리 구비형 CMOS 메모리 PS/2 시스템은 자신의 하드웨어 구성을 기억할 수 있다. 이 구성은 확장 장치들의 ID와 이 확장 장치들이 나머지 시스템과 관련하여 기능하는 방법을 포함한다. 마이크로-채널용으로 설계된 모든 확장 카드는 고유의 식별 번호(a unique identifying number)를 갖는다. 시스템이 부팅될 때, PS/2 시스템은 인스톨된 선택사양(installed options)들을 자신의 비휘발성 메모리내의 정보와 비교하여 변동사항들을 검출함으로써 자신의 설정의 보전성을 보장한다. 설정 파일들은 참조 디스켓(a Reference Diskette)을 사용한 구성 프로시쥬어동안 자동으로 파일 시스템에 통합된다. 모델 70 및 80과 같은 몇몇 IBM PS/2 모델들에서, 참조 디스켓은 컴퓨터 시스템에 장착되어 시스템 구성 정보를 저장하는 플로피 디스켓을 포함한다. PS/2 시스템들의 구성 프로시쥬어가 꽤 단순하고 수행하기 쉽지만, 참조 디스켓은 손쉽게 또는 편리하게 근방에 저장되어야 한다. 그러나 최종 시스템 구성으로부터 얼마의 시간이 경과한 후에는 참조 디스켓을 잃어버리거나 둔 곳을 잊어버릴 수 있다. 따라서, DASD상에 참조 디스켓의 복사본을 저장해 두는 것이 바람직하게 된다.

아놀드(Arnold) 등의 미국 특허 제 5,128,995 호에는 DASD의 시스템 부분으로부터 시스템 참조 디스켓 이미지(image)를 로딩하는 장치가 개시되며, 여기서 DASD는 부트 레코드(a boot record), BIOS 이미지 및 시스템 참조 디스켓을 저장하기 위한 보호 영역을 갖는다. DASD의 일부를 보호하는 이유는 BIOS의 감염 및 손상을 방지해야 하는 필요로부터 발생한다. 어떤 시스템이 동작하는 동안, 예를 들면, 프로세서가 오퍼레이팅 시스템의 제어하에 있거나 애플리케이션을 실행중일 때, DASD 제어기는 이 피보호 영역을 무시하도록 구성된다.

점차적으로, 퍼스널 컴퓨터 시스템은 정보 운용 네트워크(예를 들면, 근거리 통신망(LAN))를 제공하도록 함께 연결되는 추세이며, 이에 의해 다수의 정보 처리 시스템들은 정보를 교환하고, I/O 장치들을 공유하며, 특정의 하드화일(a particular hardfile) 또는 디스켓과 같은 특정의 직접 액세스 저장 장치(a direct access storage device;DASD)를 이용할 수 있다.

통상, 정보 네트워크는 '클라이언트'로서 알려진 다수의 정보 처리 시스템과, '서버'로 알려진 적어도 하나의 관리자 처리 시스템을 포함하며, 이들 모두는 동선(copper wire) 및/또는 광섬유 케이블과 같은 통신 매체를 통해 서로 접속 또는 망연결된다. 통상, 서브시스템 구성요소들(즉, 클라이언트 시스템 또는 서버 시스템)에 의한 네트워크 통신은 토큰 링(Token Ring) 또는 이더넷(Ethernet)과 같은 하나 이상

의 네트워크 통신 프로토콜들을 따르는 통신 어댑터 장치들을 통해 처리된다. 부가적으로, 다수의 무선 주파수 또는 적외선 프로토콜을 통해 서버에 접속되는 무선 이동 클라이언트가 개발되었다. 유선 LAN에 대해 언급되는 문제점들이 무선 LAN에 대해서도 본질적으로 마찬가지로 반복된다.

명백히, 망연결된 정보 운용 시스템의 주요한 장점은 다수의 정보 처리 시스템들간에 다양한 유형의 정보의 통신을 제공할 수 있는 능력이다. 네트워크내에서 통신될 수 있는 정보중에서는 시스템 자체의 상태 및/또는 구성에 관한 정보가 있다. 이 구성 정보는 다양한 방식으로 처리될 수 있는데, 예를 들면, 진단을 수행하거나 다른 망연결된 정보 처리 시스템들에게 서로의 구성 및 용량을 알리는 것이다.

지금까지 시스템 구성 정보는 대형의 망연결된 시스템들내의 장애 상태(faulty conditions)를 검출하는데 이용되었다. 예를 들어, IBM사의 시스템 390은 망연결된 시스템내에서 성능을 실시간으로 감시하도록 설계되어 있다. 장애 상태 검출시, 진단 루틴(a diagnostic routine)이 수행되어 결함있는 시스템 구성요소로의 장애 상태를 격리시킨다. 그 후, 검출된 장애 상태 및 진단 결과에 관련되는 정보는 전화 라인상의 모뎀을 통해 중앙의 스테이션으로 통보된다.

종래 시스템의 경우, 실시간 장애 감시 및 장애 통보 능력은 통상 시스템 하드웨어 아키텍처내에 및/또는 오퍼레이팅 시스템 또는 애플리케이션 소프트웨어내에 구현되었다. 불행히도, 대형 컴퓨터 시스템 네트워크에 구현된 실시간 장애 검출은 퍼스널 컴퓨터들로 이루어지는 것과 같은 보다 소형의 정보 운용 시스템에는 적합하지 않다. 이것은 부분적으로는, 기존의 사실상의 표준 오퍼레이팅 시스템(예를 들면, DOS 및 윈도우즈)이 대형 시스템에서 이용가능한 장애 분석 및 보고 능력을 지원하지 않기 때문이다. 더욱 중요한 것은, 퍼스널 컴퓨터 시스템의 현재 중앙 처리 능력으로는 실시간으로 구성 감시, 장애 검출 및 보고를 실행하는 것이 불가능하다는 것이다.

그러나 퍼스널 컴퓨터와 같은 소형 정보 처리 시스템을 포함하는 정보 운용 네트워크를 통해 시스템 관련 정보를 통신할 필요는 여전히 존재하며, 이들 퍼스널 컴퓨터의 오퍼레이팅 시스템 및 CPU 능력은 실시간으로 네트워크를 통한 시스템 구성 정보의 통신을 지원하지 못한다.

이러한 요구는, 정보 운용 네트워크가 대응하는 번호의 오퍼레이팅 시스템 제어하에 동작하는 다수의 정보 처리 시스템을 갖고, 각 정보 처리 시스템은 사전결정된 시스템 구성을 갖는 네트워크에서 다루어졌다. 이 처리 시스템은 오퍼레이팅 시스템을 로딩하기 전 초기 마이크로코드 로드(an initial microcode load;IML) 기간동안 사전결정된 시스템 구성에 기초해서 시스템 구성의 변화를 검출하는 검출기를 포함한다. 이 처리 시스템은 또한 오퍼레이팅 시스템을 로딩하기 전 시스템 구성의 변화 검출시에 네트워크를 통해 시스템 구성 정보를 통보하는 통신 요소들을 포함한다.

정보 운용 네트워크는 네트워크를 통한 통신을 감시하는 모니터와, 네트워크를 통해 통신된 시스템 구성 정보를 수신하는 수신기를 갖는 관리자 정보 처리 시스템을 더 포함할 수 있다.

시스템 구성 정보는, 시스템 구성의 변화가 사용자에 의해 활성화된 경우에 정보 처리 시스템에 대한 신원확인(identification)을 포함할 수도 있다. 관리자 정보 처리 시스템은 사용자로 하여금 시스템 구성을 변경할 수 있게 허용하거나 또는 허용하지 않는 허가 신호(an authorization signal)를 제공한다.

이제 도 1a를 참조하면, 본 발명에 따른 정보 운용 네트워크(150)내에서 동작할 수 있는 퍼스널 컴퓨터 시스템(100)이 도시된다. 퍼스널 컴퓨터 시스템(100)은 적절한 용기(103)를 갖는 시스템 유닛(102), 출력 장치 또는 모니터(104)(예를 들면, 종래의 비디오 디스플레이), 키보드(110)와 같은 입력 장치, 선택사양적인 마우스(112) 및 프린터(114)와 같은 선택사양적인 출력 장치를 포함한다. 마지막으로, 시스템 유닛(102)은 디스켓 드라이브(108)(도시되지 않은 디스켓과 함께 동작함) 및 하드화일로도 알려진 직접 액세스 저장 장치(DASD)(106)와 같은 하나 이상의 대용량 저장 장치들을 포함할 수 있다.

도 1b를 참조하면, 정보 운용 네트워크(150)가 도시된다. 정보 운용 시스템(150)은 다수의 정보 처리 시스템(102, 102b)을 포함하며, 그중 하나 이상은 도 1a에 도시된 퍼스널 컴퓨터 시스템일 수 있다. 처리 시스템(102)은 본 정보 운용 네트워크(150)내에서 관리자 처리 시스템으로서 동작하는 서버(a server)이다. 처리 시스템(102b)은 클라이언트 시스템을 구성한다. 통상, 클라이언트 시스템들(102b)은 DASD(106)를 포함하지 않는 점(이들 시스템(102b)은 '무매체 클라이언트(medialess clients)'라 한다)을 제외하고는 처리 시스템(102)과 동일하다. 이들 정보 처리 시스템(102, 102b)은 잘 알려진 방식으로 서로간에 망연결되어 케이블(160)에 의해 정보 운용 네트워크(150)를 통해 정보 신호들을 전달한다.

동작시에, 정보 처리 시스템(102, 102b)은 초기 마이크로코드 로드(IML) 기간 후에 적절히 로딩된 IBM사의 OS/2 오퍼레이팅 시스템 또는 DOS 오퍼레이팅 시스템과 같은 오퍼레이팅 시스템의 제어하에 동작한다. 오퍼레이팅 시스템은 통상 IML 기간중에 시스템 메모리내로 로딩된 BIOS를 이용한다. BIOS는, 프로그래머 또는 사용자가 특정 하드웨어 장치에 대한 완전한 동작 지식이 없어도 자신의 기기를 프로그램할 수 있도록 하드웨어 장치들과 오퍼레이팅 시스템 소프트웨어간에 인터페이스를 제공한다. 예를 들어, BIOS 디스켓 모듈은 프로그래머로 하여금 디스켓 드라이브 하드웨어에 대한 완전한 지식이 없이도 디스켓 드라이브를 프로그램할 수 있게 한다. 따라서, 상이한 회사들에 의해 설계 및 제조된 다수의 디스켓 드라이브가 시스템(100)내에서 사용될 수 있다. 또한, IML 기간동안에 POST 프로그램이 로딩되며, POST 프로그램은 파워 온시 시스템 하드웨어에 대한 자기 테스트를 수행하여 시스템 구성을 결정한다. POST는 그 시스템 구성을 비휘발성 RAM(NVRAM)과 같은 저장 장치에 저장한다. 이와 같이 하여, POST는 사전결정된 시스템 구성을 현재의 시스템 구성과 비교함으로써 시스템 구성의 변화를 검출할 수 있다. BIOS 및 POST는 본 명세서에서 참조로 인용된 IBM 퍼스널 시스템/2 및 퍼스널 컴퓨터 BIOS 인터페이스 기술 참조 1991 에 더욱 명백히 정의되어 있다.

일체화된 플래너

도 2를 참조하면, 정보 처리 시스템(102, 102b)의 일체화된 플래너(200)에 대한 블록도가 도시된다. 플래너(200)는 인쇄 회로 기판(a printed circuit board;PCB)(201)을 포함하는데, 그 위에 I/O 슬롯들을 갖는 다수의 입/출력 버스 커넥터(232), 버스 제어 유닛(214)의 제어하에 고속 CPU 로컬 버스(210)를 통

해 메모리 제어 유닛(256)에 접속되는 프로세서(202)가 탑재 또는 접속된다. 유닛(256)은 또한 휘발성 랜덤 액세스 메모리(RAM)(264)와 같은 주 메모리에 접속된다. 프로세서로는 인텔 80386, 인텔 80486 등과 같은 어떤 적절한 프로세서(202)도 사용될 수 있다. 시스템 파워 커넥터(205)는 PCB(201)상에 탑재되어, 시스템(100)에 필요한 전력을 공급하는 파워 유닛(도시되지 않음)에 접속된다.

CPU 로컬 버스(210)(어드레스, 데이터 및 제어 성분들을 포함함)는 프로세서(202), 선택사양적인 산술 코프로세서(204), 선택사양적인 캐쉬 제어기(206) 및 선택사양적인 캐쉬 메모리(208)의 상호접속을 위해 제공된다. 이 CPU 로컬 버스(210)에는 시스템 버퍼(212)가 또한 결합된다. 시스템 버퍼(212) 자체는 어드레스, 데이터 및 제어 성분들로 이루어지는 시스템 버스(216)에 접속된다. 이 시스템 버스(216)는 시스템 버퍼(212)와 I/O 버퍼(228)간에 연장된다. 시스템 버스(216)는 또한 버스 제어 유닛(214) 및 직접 메모리 액세스(a direct memory access;DMA) 제어 유닛(220)에 접속된다. DMA 제어 유닛(220)은 중앙 중재기(a central arbiter)(224) 및 DMA 제어기(222)를 포함한다. I/O 버퍼(228)는 시스템 버스(216)와 I/O 버스(230)간의 인터페이스를 제공한다. 오실레이터(207)는 도시된 바와 같이 컴퓨터 시스템(100)에 적절한 클럭 신호를 제공하기 위해 접속된다. 당 분야에 숙련된 자라면, 본 바람직한 실시예가 당 분야에 잘 알려진 IBM PS/2 컴퓨터 시스템의 마이크로 채널 버스상에 구현되었지만, 다른 버스 아키텍처도 또한 본 발명을 이용하는데 사용될 수 있음을 알 것이다.

I/O 버스(230)에는, I/O 장치들 또는 메모리에 또한 접속될 수 있는 어댑터 카드를 수용하기 위한 슬롯들(232)을 갖는 다수의 I/O 버스 커넥터가 접속된다. 편의상 두 개의 I/O 커넥터(232)가 도시되었지만, 특정 시스템의 필요에 따라 추가의 I/O 커넥터들이 용이하게 부가될 수 있다. 도시된 바와 같이, 이들 I/O 커넥터들중 하나는 잘 알려진 토큰 링 통신 어댑터 카드(231)에 접속되며, 이것은 정보 처리 시스템(102, 102b)에 네트워크 통신 능력을 제공하는 데 사용된다. 네트워크 통신 개시시에, CPU(202)는 잘 알려진 방식으로 토큰 링 어댑터 카드(231)를 활성화시켜서 입력되거나 출력되는 정보가 정보 운용 네트워크(150)를 통해 전달될 수 있게 한다. 통신 어댑터 카드(231) 자체는 네트워크(150)를 통해 정보를 전달하기 위한 송신 수단 및 수신 수단을 포함한다. 중재 버스(226)는 DMA 제어기(222) 및 중앙 중재기(224)를 I/O 커넥터(232) 및 디스켓 어댑터(246)에 결합한다. 시스템 버스(216)에는 또한, 메모리 제어기(258), 어드레스 멀티플렉서(260) 및 데이터 버퍼(262)를 포함하는 메모리 제어 유닛(256)이 접속된다. 메모리 제어 유닛(256)은 RAM 모듈(264)로서 표시된 랜덤 액세스 메모리와 같은 주 메모리에 또한 접속된다. 메모리 제어 유닛(256)은 프로세서(202)와 RAM(264)의 특정 영역간에 어드레스 매핑을 위한 로직을 포함한다. 기본적인 1메가바이트 RAM 모듈(264)을 갖는 시스템(100)이 도시되었지만, 도 2에 선택사양적인 메모리 모듈(266, 268, 270)로 표시된 바와 같이 부가의 메모리가 상호접속될 수 있다.

버퍼(218)는 시스템 버스(216)와 플래너 I/O 버스(234) 사이에 결합된다. 플래너 I/O 버스(234)는 어드레스, 데이터 및 제어 성분들을 포함한다. 플래너 I/O 버스(234)를 따라, 디스플레이 어댑터(236)(선택사양적인 디스플레이(104)를 구동하는데 사용됨), 클럭/CMOS RAM(250), 비휘발성 RAM(248)(이후 NVRAM이라 함), 직렬 어댑터(240)('직렬(serial)'이라는 의미로 사용되는 그 밖의 통상적인 용어는 '동기(synchronous)' 및 'RS232'임), 병렬 어댑터(238), 다수의 타이머(252), 디스켓 어댑터(246), 키보드/마우스 제어기(244), 인터럽트 제어기(254), 및 펌웨어 서브시스템(242)과 같은 다양한 I/O 어댑터들과 다른 주변장치 구성요소들이 결합된다. 펌웨어 서브시스템(242)은 통상 비휘발성 프로그램 저장장치(예를 들면, ROM)를 포함하는데, 이 저장장치는 단계 I POST로서 알려진 POST 및 BIOS 프로그램의 일부를 포함한다. 이후 상세히 설명되는 바와 같이, 단계 I POST는 초기의 제한된 시스템 테스팅을 위해 사용되며, 플로피 드라이브(108) 또는 하드화일(106)과 같은 외부 저장장치로부터, 초기 마이크로 코드 로드(IML) 이미지로도 알려진 단계 II POST를 로딩하기 위한 루틴들을 포함한다.

클럭/CMOS RAM(250)은 일일 시간을 계산하는데 사용된다. NVRAM(248)은 시스템 구성을 저장하는데 사용된다. 즉, NVRAM(248)은 시스템(100)의 현재 구성을 기술하는 값들을 포함하게 된다. NVRAM(248)은, 예를 들면, 어댑터 카드 초기화 데이터, 고정된 디스크 또는 디스켓 용량, 주 메모리 용량 등을 기술하는 정보를 포함한다. 또한, 이들 데이터는 구성 프로그램이 실행될 때마다 NVRAM(248)에 저장된다. 이러한 구성 프로그램은 IBM PS/2 컴퓨터 시스템과 함께 포함된 시스템 참조 디스켓상에 제공되는 통상의 구성 설정(Set Configuration) 프로그램일 수 있다. 참조 디스켓은 때때로 진단, 유지보수 또는 서비스 디스켓(a diagnostic, maintenance or service diskette)이라 지칭된다. 구성 프로그램의 목적은 이 시스템(100)의 구성을 특성화하는 값들을 사전결정하여 사전저장하는 것으로, 이들 값은 시스템으로부터 파워가 제거될 때 NVRAM(248)에 보존된다. NVRAM은 배터리 백업을 갖는 저파워 CMOS 메모리일 수 있다.

키보드/마우스 제어기(244)에는 포드 A(278) 및 포트 B(280)가 접속된다. 이들 포트 A, B는 키보드(110) 및 마우스(112)를 퍼스널 컴퓨터 시스템(100)에 접속하는데 사용된다. 직렬 어댑터(240)에는 직렬 커넥터(276)가 결합된다. 모뎀(도시되지 않음)과 같은 선택사양적인 장치가 이 커넥터(276)를 통해 시스템에 결합될 수 있다. 병렬 어댑터(238)에는 프린터(114)와 같은 장치가 접속될 수 있는 병렬 커넥터(274)가 결합된다. 디스켓 어댑터(246)에는 하나 이상의 디스켓 드라이브(108)을 부착하는데 사용되는 디스켓 커넥터(282)가 접속된다.

다른 플래너 보드

퍼스널 컴퓨터 시스템(100)의 다른 실시예에 따르면, 일체화된 플래너(200)가 플래너 보드(300) 및 프로세서 카드(400)(도 3 및 4 참조)로 대체된다. 프로세서 카드(400)는 플래너 보드(300)상에 착탈가능하게 장착되어 전기적으로 접속된다. 도 2의 구성요소 번호와 동일한 도 3 및 도 4의 동일한 구성요소 번호는 동일한 구성요소에 대응한다. 이제 도 3을 참조하면, 플래너 보드(300)는 PCB내의 배선 또는 회로들에 의해 상호접속된 다양한 구성요소들이 탑재(예를 들면, 표면 실장)되는 인쇄 회로 기판(a printed circuit board;PCB)(301)을 포함한다. 이들 구성요소는 적절하게 상업적으로 이용가능한 전기적 커넥터(302)를 포함하는데, 이 커넥터에 프로세서 카드(400)의 에지(416)가 플러깅되어 프로세서 카드(400)를 플래너 보드(300)에 착탈가능하게 탑재하여 전기적으로 접속한다. 다수의 단일 인-라인 메모리 모듈(single in-line memory module;SIAM) 커넥터(306)가 PCB(301)상에 또한 탑재되어 시스템 주

메모리 또는 RAM을 형성하는 메모리 뱅크(308A, 308B)에 접속시킨다. 하나 이상의 I/O 버스 또는 확장 커넥터(232)가 또한 PCB(301)상에 탑재되어 퍼스널 컴퓨터 시스템(100)에 부가 또는 결합될 수도 있는 다른 확장 어댑터들 및 선택사양들에 접속시킨다. 예를 들어, 고정된 디스크 드라이브(106)는 소형 컴퓨터 시스템 인터페이스(a small computer system interface;SCSI) 디스크 제어기를 갖는 어댑터 카드(231)에 접속된다. 어댑터 카드(231)는 I/O 버스 또는 확장 커넥터(232)에 접속된다. 바람직하게, 각각의 커넥터(232)는 앞서 언급된 마이크로 채널 아키텍처에 적합한 유형의 상업적으로 이용가능한 커넥터이다.

플래너 보드(300)상에는 또한 키보드 및 마우스 커넥터(278, 280)에 접속되는 키보드/마우스 제어기(244) 및 인터럽트 제어기(254), 디스켓 커넥터(282)에 접속된 디스켓 제어기 또는 어댑터(246), 및 다양한 I/O 장치들이 시스템에 접속될 수 있게 하는 직렬 및 병렬 커넥터(276, 274)에 접속된 직렬 및 병렬 어댑터(240, 238)가 탑재된다. 시스템 파워 커넥터(205)는 PCB(301)상에 탑재되어 시스템에 필요한 파워를 공급하는 파워 유닛(도시되지 않음)에 접속시킨다. 비휘발성 메모리(NVRAM)(248) 및 일일 시간 클럭/CMOS RAM(250)이 또한 PCB(301)상에 탑재된다. PCB(301)는 또한 타이밍 신호를 제공하기 위한 다양한 발진기들(도시되지 않음)과, 잘 알려진 방식으로 회로의 부분들을 격리시키기 위한 버퍼들(342, 344)(도시되지 않음)을 탑재하였다.

PCB(301)의 배선은 도면중에 도시된 바와 같이 다양한 구성요소들을 상호접속하고, 세 개의 그룹, 즉, 메모리 버스(310)(라인(324-338)을 포함함), 채널 버스(312)(어드레스 버스(322), 데이터 버스(320) 및 제어 버스(318)를 포함함), 및 인터럽트 라인들(314, 316)을 포함하는 각종 신호 라인들로 나뉘어지며, 이들 모두는 커넥터(302, 416)를 통해 PCB(401)상의 대응하는 배선에 접속된다. 버스(312)로부터 플래너 기능 버스(319)가 탭(tap)된다.

프로세서 카드

도 4를 참조하면, 플래너 보드(300)상에 착탈가능하게 탑재되는 프로세서 카드(400)가 도시된다. 프로세서 카드(400)는 인쇄 회로 기판(PCB)(401)을 포함하는데, 이 PCB(401)상에는 프로세서(202), 선택사양적인 산술 프로세서(204), 선택사양적인 캐쉬 제어기(206), 선택사양적인 캐쉬 메모리(208), 직접 메모리 액세스(DMA) 제어 유닛(220), 버스 제어 유닛(214), 메모리 제어 유닛(256), 펌웨어 서브시스템(242) 및 패리티 체킹 유닛(402, 404)을 포함하는 다수의 상업적으로 이용가능한 구성요소들이 탑재(예를 들면, 표면 실장)된다. 프로세서(202)는 바람직하게, 32비트 데이터 경로를 갖고 32비트 어드레싱 능력을 제공하는 인텔 80486과 같은 고성능형이다. 물론, 인텔 80386 등의 프로세서가 사용될 수 있다. 나머지 구성요소들은 프로세서와의 호환성을 위해 통상의 방식으로 선택된다. 다수의 버퍼(406, 408, 410, 412, 414)는 도시된 바와 같이 접속된다. 버퍼는 회로들간에 선택적인 분리 또는 접속을 제공하여 상이한 부분들이 동시에, 예를 들면, I/O 유닛과 주 메모리(308A, 308B) 사이에 다른 데이터가 전송되고 있는 동안 프로세서(202)와 캐쉬 메모리(208) 사이에 데이터를 전송하는데 이용될 수 있게 한다. 이상의 구성요소들 모두는 에지 커넥터(416)에서 종단되는 PCB(401)내의 인쇄된 배선 회로에 의해 적절히 서로에게 전기적으로 접속된다. 에지 커넥터(416)는 도 3에 도시된 플래너 보드(300)상의 에지 커넥터(302)에 플래깅가능하므로, 플래너 보드(300)와 프로세서 카드(400)는 전기적 및 기구적으로 상호접속가능하다. 이와 같이 하여, 제각기 연관된 프로세서 유형을 갖는 다양한 버전의 프로세서 카드(400)가 플래너 보드(300)에 플러깅될 수 있다.

PCB(401)의 배선 회로는 데이터, 어드레스 및 제어 라인들(420, 422, 424)을 갖는 로컬 버스(418)를 포함하며, 이들 라인은 제각기 도 4에 도시된 바와 같이 프로세서(202)를 선택사양적인 산술 코프로세서(204), 선택사양적인 캐쉬 제어기(206) 및 선택사양적인 캐쉬 메모리(208)와 제각기 상호접속한다. 나머지 회로 라인들은 통상 인터럽트 라인(316), 채널 버스 라인(312) 및 메모리 버스 라인(310)을 포함한다. 채널 버스 라인들(312)은 제각기 제어, 데이터 및 어드레스 버스 라인들(318, 320, 322)을 포함한다. 메모리 버스 라인들(310)은 다중화된 메모리 어드레스 라인(324, 332), 메모리 뱅크(308A, 308B)용 행 어드레스 스트로브(row address strobe;RAS) 라인(328, 336), 열 어드레스 스트로브(column address strobe;CAS) 라인(338), 데이터 버스 A 및 B 라인(326, 334), 및 패리티 체크를 통한 에러 체킹 또는 ECC 체킹에 사용하기 위한 라인(330)을 포함한다. 발진기(207)는 적절한 클럭 신호를 컴퓨터 시스템(100)에 제공하기 위해 도시된 바와 같이 접속된다. 도시를 간략하게 하기 위해 특정의 각종 라인들, 예를 들면, 리셋, 그라운드, 파워-온 라인들 등은 도 2 내지 4로부터 생략되었다.

보드(300) 및 카드(400)를 갖는 퍼스널 컴퓨터 시스템(100)의 정상 동작중에, 카드(400)는 보드(300)에 전기적 기구적으로 접속되며, 통상적으로 보드(300)에 사실상 수직인 평면에 놓인다.

반복하면, 시스템 펌웨어는 파워-온 셀프 테스트 프로그램(POST) 및 기본 입/출력 시스템 프로그램(BIOS)을 포함한다. BIOS는 또한 호환성 BIOS(혹은 CBIOS) 및 고급 BIOS(혹은 ABIOS)를 포함한다. POST는 시스템이 처음 파워-온될 때 실행할 인스트럭션들의 세트이다. 현재의 시스템 구성이 결정된 때에 퍼스널 컴퓨터 시스템(100)을 초기화시키기 위해 POST를 실행시키는 것이 중요하다. BIOS는 프로세서(202)와 I/O 장치들간에 데이터 및 제어 인스트럭션들의 전송을 용이하게 하는 인스트럭션들의 세트이다.

정보 처리 시스템(예를 들면, 102B)이 무매체인 경우(DASD 또는 플로피 디스크를 갖지 않는 경우)에, 원격 초기 프로그램 로드(a Remote Initial Program Load;RIPL)가 통신 어댑터 카드(231)를 통해 수행된다. 이 카드는, 예를 들면, 커넥터들(232)중 하나에 접속되며, 잘 알려진 방식으로 네트워크 서버(102)로부터 오퍼레이팅 시스템을 부팅시킬 수 있게 한다.

POST는 부팅 장치를 찾아서 부트 레코드를 로딩하는 부트스트랩 프로그램(a bootstrap program)을 포함한다. 통상, 부팅 장치는 하드화일(106) 또는 디스켓 드라이브(108)이며, 또는 무매체 클라이언트의 경우 부팅 장치는 서버(120)이다. 디스켓 드라이브(108)는 동작하는데 부팅 디스켓 또는 오퍼레이팅 시스템 디스켓(매체)(도시되지 않음)을 필요로 한다. POST가 부팅 장치로부터 부트 레코드를 로딩하는데 성공하면, POST는 제어를 부트 레코드로 넘겨주고 POST 부트스트랩 프로그램의 동작을 완료한다. 부트 레코드가 로딩될 수 없고 RPL 어댑터가 존재하는 경우, POST는 RPL 프로그램에게 제어를 넘겨준다. RPL

프로그램이 존재하지 않는 경우, POST는 부트 소스가 필요함을 표시하여 사용자를 환기시킨다. CBIOS는 컴퓨터의 부트스트랩 동작에 필수적이다. CBIOS는 하드화일(106) 및 디스켓 드라이브(108)에 대한 액세스를 포함하는 다수의 서비스를 제공한다.

정보 운용 시스템(102, 102B)은 초기 마이크로코드 로드(IML)로 잘 알려진 2단계 POST를 실행하며, 이에 관하여는 본 명세서에서 참조로 인용된 미국 특허 제 5,128,995 호에 충분히 기술되어 있다. IML에 따르면, 단계 I POST동안 프로세서는 펌웨어 서브시스템(242)의 내용을 로딩하여, 디스플레이 장치와 같은 특정 I/O 장치들과 특정의 메모리 장치 어드레스에 대한 최소 체킹을 포함하는 커맨드를 실행한다. 그 후, 단계 II POST 동안, 프로세서는 IML 이미지를 사용하여 시스템 테스팅을 완료한다. IML 이미지는 POST를 완료하고 프로세서 제어를 오퍼레이팅 시스템으로 전환시키는 인스트럭션들을 포함한다. POST 동안, 현재 시스템 구성이 결정되며, 사전결정되어 NVRAM(248)에 사전저장된 시스템 구성 정보에 비교된다. IML 이미지는 시스템 프로세서에 따라 특정되며, 저장 장치의 시스템 파티션(a system partition)에 저장된다. 저장 장치가 플로피 드라이브(108)내의 디스켓인 경우, 시스템 파티션은 디스켓의 일부를 점유한다. 이 시스템 파티션에 저장된 IML 이미지는 시스템 하드웨어를 설정 및/또는 진단하기 위한 루틴들을 포함하는 참조 이미지(a reference image)는 물론 POST의 일부를 포함한다. 참조 이미지는 POST가 구성 변화로 인한 에러를 검출했을 때 로딩되어 실행된다. 참조 이미지는 또한 사전정의된 키 시퀀스(a predefined key sequence)에 응답하여 로딩될 수도 있다. 저장 장치가 하드화일(106)인 경우, 대단히 중요한 IML 이미지가 부주의하게 손상되는 것을 회피하기 위해 시스템 파티션은 저장 매체의 보호 영역에 저장된다. IBM사의 모델 90 및 95에서 보호되는 시스템 파티션은 저장 매체의 제일 마지막 부분에 저장되며, 3메가바이트 저장 용량을 점유한다.

본 발명에 따르면, IML 기간중 시스템 구성에 있어서의 변화 검출시 정보 운용 네트워크(150)의 정보 처리 시스템(102, 102B)은 통신 어댑터 카드(231)를 활성화시켜서 오퍼레이팅 시스템을 로딩하기 전에 네트워크를 통해 특정의 시스템 구성 정보를 전송한다.

도 5를 참조하면, 본 발명의 목적을 달성하기 위해 수행되는 동작 단계들이 흐름도(500)로 예시된다. 시스템이 파워 온되면(블록(501)), 정보 처리 시스템은 IML 기간동안 POST를 수행한다(블록(502)). 그리고 나서, 처리 시스템은 시스템 구성 변경이 시도되었는지의 여부를 판정한다. 시스템 구성의 변경은 시스템 메모리를 증가시키거나 감소시키는 것과 같이 하드웨어 구성을 변경함으로써 발생될 수도 있다. 이와 달리, 사용자가 사전정의된 키를 두드리는 시퀀스를 실행함으로써 시스템 구성 변경을 요구할 수 있는 경우에 사용자에 의해 구성 변경이 표시될 수도 있다. 우선, 정보 처리 시스템은 POST를 통해 IML 기간 동안에 하드웨어 구성 변화가 있었는지의 여부를 판정한다(블록(503)). 이러한 판정은 현재 시스템 구성을 NVRAM(248)에 저장된 사전결정된 시스템 구성과 비교함으로써 수행된다. 변화하지 않았으면, 사용자에 의한 구성 변경이 수행되었는지의 여부가 판정된다(블록(504)). 이러한 판정은 잘 알려진 방식으로 사용자에 의한 사전정의된 키 시퀀스의 실행을 검출함으로써 수행된다. 사용자에 의해 구성 변경이 초기화되지 않았으면, 오퍼레이팅 시스템이 로딩된다(블록(505)).

그러나 POST가 시스템의 하드웨어 구성이 변경된 것으로 판정하면, 특정 시스템 구성 정보가 수집되어 오퍼레이팅 시스템을 로딩하기 전에 네트워크를 통해 전달된다(블록(506)). 이 시스템 구성 정보는 다른 것들 중에서도 하나 이상의 어댑터 카드 오류에 대한 에러 코드, NVRAM, CMOS 및 BIOS 데이터 영역 등을 포함할 수 있다. 그 후, 시스템 파티션으로부터 참조 이미지가 로딩되어, NVRAM(248)에 현재의 시스템 구성을 저장하도록 실행된다.

POST가 사용자에 의한 시스템 구성 변경이 시도된 것으로 판정하면, 시스템 파티션 액세스가 허용되는지의 여부에 대한 판정이 수행된다(블록(507)). 허용되지 않으면, 시스템 구성 정보가 수집되어 네트워크를 통해 전달되며(블록(508)), 계속해서 오퍼레이팅 시스템이 로딩된다(블록(509)). 그러나, 시스템 파티션 액세스가 허용되면, 이러한 액세스가 관리자 정보 처리 시스템, 예를 들면, 서버(102)로부터의 허가를 요구하는지에 관한 판정이 수행된다(블록(510)). 요구하지 않으면, 참조 이미지가 로딩되고 실행되어 사용자로 하여금 시스템 구성을 변경할 수 있도록 한다. 시스템 파티션 액세스가 허가를 요구하면, 변경될 시스템 구성의 ID를 포함하는 시스템 구성 정보 변경 요구가 네트워크 통신 어댑터 카드(231)를 통해 서버(120)에게 전달된다(블록(511)). 서버(120)가 시스템 파티션에 대한 액세스를 허가하면, 참조 이미지가 로딩되어 실행된다(블록(512, 513)). 서버(120)가 액세스를 허가하지 않으면, 오퍼레이팅 시스템이 로딩된다(블록(512, 505)).

도 6을 참조하면, 블록(506, 508)에서 수행되는 단계들의 보다 상세한 흐름도(600)가 도시된다. 시스템 구성 변경(하드웨어 변경 또는 사용자에 의한 구성 변경 초기화)의 검출시, 네트워크 통신 어댑터 카드(231)가 활성화된다(블록(601)). 이 때, 정보 처리 시스템은 NVRAM(248)으로부터 사전결정된 시스템 구성을 획득하기 위해 드라이버를 로딩한다(블록(602)). 그리고 나서, 드라이버는 다수의 장치로부터 시스템의 현재 하드웨어 구성을 수집한다(블록(603)). 그리고 나서, 에러 관련 정보, 즉, 하드웨어 구성 변경의 에러 코드 또는 사용자 액세스 요구 정보가 수집된다(블록(604)). 그리고 나서, 선행하는 단계들로부터 수집된 모든 정보가 포맷된 후 통신 어댑터 카드(231) 및 네트워크를 통해 전달된다.

전달된 시스템 구성 정보는 통신 어댑터 카드(231)를 통해 관리자 처리 시스템(예를 들면, 서버(102))에 의해 수신된다. 관리자 시스템은 바람직하게는 소프트웨어 루틴 형태로 네트워크(150)상의 통신을 감시하는 감시 수단을 포함한다. 시스템 구성 정보 검출시, 관리자 시스템은 정보를 수신하여 그것을 DASD 등의 영구 저장 매체에 기록한다. 저장된 정보는 차후의 처리를 위해 바람직하게 포맷되고 시간이 기록된다.

시스템내의 각 정보 처리 시스템은 본 발명에 따라 전달된 시스템 구성 정보를 수신하는 능력을 가질 수 있다. 이와 같이하여 네트워크 시스템에 의해 수신된 정보는 그들에게 네트워크내의 다른 시스템들의 능력을 알리고, 그렇게 함으로써, 그들의 하드웨어가 네트워크 능력을 충분히 이용하도록 적응시킬 수 있다.

알 수 있는 바와 같이, 본 발명은 오퍼레이팅 시스템 또는 어떤 다른 애플리케이션 소프트웨어를 로딩하기 전에 시스템 구성 정보를 전달할 수 있는 능력을 제공한다. 따라서, 이러한 귀중한 네트워킹 특징을

IBM 호환 퍼스널 컴퓨터와 같은 기성의 오퍼레이팅 시스템을 채용하는 정보 처리 시스템을 갖는 정보 운용 네트워크에 이용가능하게 한다. 더욱이, 본 발명은 동작개시시 필요할 때 및 오퍼레이팅 시스템을 로딩하기 전에 시스템 구성 정보가 전달될 수 있게하여 프로세서가 실시간으로 구성 감시 태스크를 수행할 필요를 제거한다.

소프트웨어 환경

개요-객체 지향형 기술

본 발명은 객체 지향형(Object Oriented:OO) 프레임워크 기술을 이용하여 구현될 수 있다. OO 프레임워크 기술 분야에 숙련된 자라면, 본 명세서의 상세한 설명중 다음 부분으로 진행하기를 바랄 수도 있다. 그러나, 프레임워크 기술에 생소한, 또는 전반적으로 OO 기술에 생소한 자라면 본 발명의 잇점 및 장점들을 가장 잘 이해할 수 있도록 이 개요 부분을 숙독해야 한다.

객체 지향형 기술과 프로시쥬얼 기술

본 발명은 특히 OO 기술에 관련되지만, 우선, 전반적으로 OO 기술이 통상의 프로세스-기반 기술(종종 프로시쥬얼 기술로 칭하여짐)과 현저히 상이함을 이해해야 한다. 양 기술은 동일한 문제를 해결하는데 사용될 수 있지만, 그 문제에 대한 궁극적 해(solution)는 항상 매우 상이하다. 이러한 차이는 프로시쥬얼 기술의 설계 초점이 전체적으로 OO 기술의 설계 초점과 상이하다는 사실로부터 발생한다. 프로세스-기반 설계의 초점은 문제를 해결하는 전체적인 프로세스에 놓여지는 반면, OO 설계의 초점은, 해를 제공하기 위해 함께 동작할 수 있는 독립적 개체 세트(a set of autonomous entities)로 문제가 나뉘어질 수 있는 방법에 놓여진다. OO 기술에서의 독립 개체를 객체(objects)라고 한다. 다른 말로 하면, OO 기술은 프로시쥬얼 기술과 현저히 상이한데, 그 이유는, OO 기술에서는 문제가 중첩된 컴퓨터 프로그램들 또는 프로시쥬어들의 계층으로 나뉘어지는 대신에 함께 동작하는 객체들의 세트들로 나뉘어지기 때문이다.

'프레임워크'라는 용어

OO 설계 분야에 숙련된 자에게 특정한 의미를 갖는 용어 및 문구들이 개발되었다. 그러나, OO 기술 분야에서 가장 부정확한 정의들중 하나가 '프레임워크(framework)'라는 단어의 정의임에 주목해야 한다. 이 프레임워크라는 단어는 여러 사람에게 각기 다른 것을 의미한다. 따라서, 두가지 제안된 프레임워크 메카니즘의 특징을 비교할 때, 이 비교가 정말 '사과 대 사과'의 비교임을 보장하는지에 주의해야 한다. 더욱 명백하게 되는 바와 같이, 프레임워크라는 용어는 본 명세서에서 핵심 기능 및 확장 기능을 갖도록 설계된 OO 메카니즘을 기술하는데 사용된다. 핵심 기능은 프레임워크 구매자에 의해 수정될 수 없는 프레임워크 메카니즘의 일부분이다. 이와 달리, 확장 기능은 프레임워크 구매자에 의해 커스텀화 및 확장되도록 명시적으로 설계된 프레임워크 메카니즘의 일부분이다.

OO 프레임워크 메카니즘

통상 OO 프레임워크 메카니즘은 OO 해로서 적절히 특성화될 수 있지만, 그럼에도 불구하고 프레임워크 메카니즘과 기본 OO 해(a basic OO solution) 사이에는 기본적인 차이가 있다. 그 차이는, 프레임워크 메카니즘이 해의 측정 측면들의 커스텀화 및 확장을 허용하고 촉진하는 방식으로 설계된다는 것이다. 즉, 프레임워크 메카니즘은 그 문제에 대한 바로 그 해 이상이다. 이들 메카니즘은 시간이 경과함에 따라 변화하는 개별화된 요건들을 다루도록 커스텀화되고 확장될 수 있는 동적 해(a living solution)를 제공한다. 물론, 프레임워크 메카니즘의 커스텀화/확장 성질은 구매자(본 명세서에서 프레임워크 소비자로 언급됨)에게 매우 가치있는 것인데, 그 이유는, 프레임워크를 커스텀화하고 확장하는 비용이 기존의 해를 대체하거나 개정하는 비용보다 훨씬 적기 때문이다.

따라서, 프레임워크 설계자가 특정 문제를 해결하고자 착수할 때, 그들은 단지 개개의 객체와 이들 객체가 상호관련되는 방법을 설계하는 것 이상을 한다. 그들은 또한 프레임워크의 핵심 기능(즉, 프레임워크 소비자에 의해 커스텀화 및 확장 불가능한 프레임워크의 일부분) 및 프레임워크의 확장 기능(즉, 참재적인 커스텀화 및 확장이 가능한 프레임워크의 일부분)을 설계한다. 결국, 프레임워크 메카니즘의 궁극적인 가치는 객체 설계의 성질에 있을뿐 아니라, 프레임워크의 어느 측면들이 핵심 기능을 나타내고 어느 측면이 확장 기능을 나타낼 것인지를 포함하는 설계 선택(design choices)에 있다.

ZAF-예시적인 프레임워크 메카니즘

당 분야에 숙련된 자라면 프레임워크 설계가 필연적으로 한데 꼬인 반복적 프로세스임을 알 것이지만, 극히 단순한 프레임워크 메카니즘에 대한 예시적인 설계 선택은 이하에서 설명된다. 그러나 이것은, 본 발명의 잇점 및 장점들을 보다 잘 이해하고 알 수 있도록 프레임워크 메카니즘을 예시하고 가장 잘 설명하기 위해 본 명세서에서 사용되고 있는 단지 예시적인 프레임워크임을 이해해야 한다.

프레임워크 설계자들은 문제 영역(problem domain)으로 불리워지는 객체들을 선택함으로써 프레임워크 메카니즘에 어떤 객체가 필요한지를 결정한다. 문제 영역은 가까이에 있는 특정 문제의 추상적 개념(an abstract view)이다. 본 예시적인 프레임워크 메카니즘에 대해 선택된 예시적인 문제 영역은 동물원 관리(zoo administration)의 문제이다. 이 특정 문제는 동물원의 동물들을 돌보고 사육하는 데 있어 동물원 사육사들을 돕는 메카니즘을 설계하는 문제이다. 본 동물원 관리 프레임워크(a Zoo Administration Framework:ZAF) 실시예에서, OO 프레임워크 설계자는 동물원의 문제 영역을 감시하여, 어떤 ZAF가 동물원 사육사들과 동물들간의 관계를 나타낸(즉, 동물원 사육사들이 동물들을 돌보는 방법을 나타내기 위한) 메카니즘을 필수적으로 포함함을 판정한다. 프레임워크 설계자는 또한 동물원 동물들이 통상 우리(cages), 축사(pens), 수조(tanks) 및 다른 종류의 억제 장치들내에서 생활함을 알 수 있을 것이다. 따라서, 본 프레임워크 설계자는, 프레임워크가 이들 모든 기본적인 개체들 및 그들의 관계를 나타낸 메카니즘을 포함해야 할 것이라는 지식을 갖고 시작하게 된다.

설계 프로세스를 시작하기 위해, 프레임워크 설계자는 아마도 소위 카테고리 도면(a category diagram)으로 불리우는 것으로 시작하게 될 것이다. 카테고리 도면은 고급 프레임워크 메카니즘과 이들 메카니

증이 서로에게 관련되는 방법을 기술하는데 사용된다. 도 7은 본 실시예의 프레임워크 ZAF에 대한 카테고리 도면이다. 카테고리 도면에서 각 메카니즘은 특정의 기능을 수행하는 그룹화된 객체들을 나타낸다. 예시를 목적으로, 프레임워크 설계자가, ZAF는 4개의 고급 메카니즘, 즉, 동물원 관리 메카니즘, 동물원 사육사 메카니즘, 동물 메카니즘, 및 억제 장치 메카니즘으로 이루어져야 한다고 결정했다고 가정하자.

도 7에 도시된 바와 같이, 동물원 관리 메카니즘은 동물원을 관리하기 위해 동물원 사육사 메카니즘을 사용하도록 설계되었다. 따라서, 이 동물원 관리 메카니즘은 동물원 사육사 메카니즘과 사용 관계(a using relationship)를 갖는다고 말하여진다.

설명된 바와 같이, 동물원 관리 메카니즘은 ZAF의 종합적인 제어에 대한 책임을 갖도록 설계되었다. 따라서, 동물원 관리 메카니즘은 동물원 사육사 메카니즘의 동작을 스케줄링할 책임이 있다. 또한, 본 프레임워크 설계자는 동물원 관리 메카니즘이 ZAF의 핵심 기능이 되도록 설계하였으며, 이것은 잠재적으로 커스텀화 및 확장되지 않도록 설계되었음을 의미함에 주목하자. 카테고리 박스내의 C는 이러한 사실을 표시한다. 더욱이, 동물원 관리 메카니즘과 동물원 사육사 메카니즘간의 사용 관계는 프레임워크 소비자에 의한 궁극적인 커스텀화에 이용되지 않도록 또한 설계되었음에 주목하자.

동물원 사육사 메카니즘은 동물원의 동물들을 돌보고 사육하는데 전반적으로 책임을 지도록 설계되었다. 따라서, 자신의 태스크를 수행하기 위해 동물 메카니즘과 억제 장치 메카니즘을 사용한다. 그러나 동물 관리 메카니즘의 설계와 달리, 프레임워크 설계자는 동물 사육사 메카니즘을 확장 기능으로 설계하였으며, 이것은 또한 동물원 사육사 메카니즘이 프레임워크 소비자에 의해 수정 및/또는 확장될 수 있도록 설계되어 미래의 돌봄 및 사육 요건을 다룰수 있음을 의미한다. 이러한 사실은 동물원 사육사 메카니즘 카테고리 박스내에 E로 표시된다.

본 프레임워크 설계자는 동물원의 동물들과 동물원의 사육사간의 상호작용을 동물 측면에서 나타내도록 동물 메카니즘을 설계하였다. 동물원내 동물들의 수는 규칙적으로 변화하는 것이므로, 동물 메카니즘은 확장 기능으로서 유사하게 설계되었다. 억제 장치 메카니즘은 축사, 수조 및 우리와 같은 개개의 억제 장치를 나타냄으로써 동물원 사육사 메카니즘과 상호작용한다. 동물 메카니즘과 마찬가지로, 억제 장치 메카니즘은 미래의 커스텀화 및 확장 요건들을 처리할 수 있도록 확장 기능으로서 설계되었다. 그러나, 여기서, 동물원 사육사 메카니즘, 동물원 동물 메카니즘 및 억제 장치 메카니즘이 모두 확장 기능으로서 설계되었더라도 이들 메카니즘간의 관계는 ZAF의 핵심 기능으로 설계되었음에 유의하자. 즉, 동물원 사육사 메카니즘, 동물원 동물 메카니즘 및 억제 장치 메카니즘에 대해 ZAF의 소비자 호환성을 제공하는 것이 바람직하지만, 서로에 대해 이들 메카니즘이 관계하는 방법을 ZAF의 소비자들이 변화시킬 수 있도록 하는 것은 바람직하지 않다.

본 프레임워크 설계자는 다음으로 도 7에 도시된 메카니즘들을 형성하는 클래스 및 관계를 설계하게 된다. 클래스는 유사한 객체들로 이루어진 세트의 정의이다. 따라서, 클래스는 객체들의 추상 개념들로서 또는 객체 유형의 정의로서 생각할 수 있다. 컴퓨터 시스템의 관점으로부터, 단일 객체는 데이터와 이 데이터에 대해 컴퓨터 시스템에 의해 수행되는 연산 또는 그룹화된 연산들의 캡슐화된 세트를 나타낸다. 사실, 안전한 컴퓨터 시스템에서, 객체에 의해 제어되는 정보에 대한 유일한 액세스는 객체 자체에 의해서이다. 이 때문에 객체내에 포함된 정보를 객체에 의해 캡슐화되었다고 말한다.

각 클래스 정의는 객체에 의해 제어되는 정보를 정의하는 데이터 정의들과 각 객체가 제어하는 데이터에 대해 객체들에 의해 수행되는 연산 또는 연산들을 정의하는 연산 정의들을 포함한다. 즉, 클래스 정의는 정의된 데이터에 대해 수행되는 연산 또는 연산들의 세트를 정의함으로써 객체가 다른 객체에 대해 작용하고 반작용하는 방법을 정의한다. (이들 연산은 때때로 메쏘드(method), 메쏘드 프로그램(method programs) 및/또는 요소_함수(member_functions)로 불림에 주의하자.) 이들 정의된 연산(들)과 데이터가 합쳐졌을 때 객체의 동작(behaviour)이라고 말한다. 본질적으로, 이 때 클래스 정의는 자신의 구성 요소 객체 또는 객체들의 동작을 정의한다.

도 8은 본 프레임워크 설계자들이 ZAF에 대해 설계한 기본적 클래스들을 도시하는 OO 클래스 도면이다. 각각의 클래스 표시는 도 7상에 도시된 메카니즘에 대한 그의 관계를 포함한다. 예를 들어, 동물원 사육사 클래스가 '동물원 사육사 메카니즘으로부터'인 것으로 표시된 것을 볼 수 있다. ZAF의 기본적인 클래스들로는, 동물원 관리 메카니즘의 일부인 동물원 관리자 클래스, 또한 동물원 관리 메카니즘의 일부인 동물원 사육사 등록 클래스, 동물원 사육사 메카니즘의 일부인 동물 등록 클래스, 동물원 사육사 메카니즘의 또한 일부인 동물원 사육사 클래스, 동물원 사육사 메카니즘의 또한 일부인 억제 장치 등록 클래스, 동물 메카니즘의 일부인 동물 클래스, 및 억제 장치 메카니즘의 일부인 억제 장치 클래스가 포함된다.

또한, 클래스들간의 관계는 궁극적으로 ZAF의 소비자에 의한 수정이 불가능하도록 ZAF의 핵심 기능으로서 설계되었음에 유의해야 한다.

동물원 관리자 클래스는 ZAF의 종합적인 제어를 책임지는 객체의 정의이다. 또한, 모든 OO 클래스들은 문제에 대해 해를 제공하기 위해 상호작용하는 객체들을 정의한다. 그러나 미래의 요건들을 다루도록 커스텀화 및/또는 확장될 수 있는 동적 해를 제공하기 위해 프레임워크 메카니즘의 객체들이 어떻게 설계되었는지를 이해할 수 있는 방법은 클래스 정의들의 특성을 탐구함으로써이다.

동물원 관리 클래스는 동물원사육사 등록 클래스와 사용 관계를 갖도록 설계되었다. 본 프레임워크 설계자는, ZAF의 소비자들이 동물원 관리 클래스 및 동물원 등록 클래스 정의들의 구성요소인 객체들의 동작을 수정할 수 있게 해서는 안된다고 판단하였으므로, 이들 클래스를 ZAF의 핵심 기능이 되도록 설계하였다. 동물원 사육사 클래스와 '참조에 의한 포함 관계(a contains by reference ralationship)'라고 불리우는 것을 갖는 동물원 사육사 등록 클래스는 단순히 모든 동물원 사육사 객체들에 대한 컨테이너(a container)인 객체를 정의하는 클래스이다. 따라서, 동물원 사육사 등록 클래스는 리스트_동물원_사육사() 연산(a list_zoo_keepers() operation)에 대한 정의를 포함한다. 이후 설명되는 바와 같이, 이 연산은 이러한 리스트를 요구하는 다른 객체들에게 동물원 사육사 객체들의 리스트를 제공한

다.

도 9는 동물원 관리자 클래스의 하위 레벨 도면을 도시한다. 이러한 객체 유형, 즉, 관리자 객체는 ZAF의 종합적인 제어를 담당하므로, 동물원 관리자 클래스는 동물원 관리를 위한 태스크를 수행하는 연산들을 포함하도록 설계되었다. 클래스 정의는 다음 5개 연산들, 즉, 5_분_타이머(), 추가_동물(), 추가_억제_장치(), 추가_동물원_사육사(), 및 개시_동물원_관리() 연산들을 포함한다.

개시_동물원_관리() 연산은 ZAF를 개시시킨다. 즉, 사용자 또는 시스템 관리자는 개시_동물원_관리() 연산으로 상호작용하여 ZAF를 통해 동물원의 관리를 개시하게 된다. 본 프레임워크 설계자는, 일단 관리가 개시되면, 개시_동물원_관리() 연산이 5_분_타이머() 연산을 초기화하도록 설계하였다. 매 5분마다 5_분_타이머() 연산은 동물원 사육사 객체들에게 밖으로 나가서 동물들을 체크하도록 지시한다. 추가/삭제_동물원_사육사 연산은 ZAF의 사용자들과 상호작용하여, 추가의 동물원 사육사들(즉, 동물원 사육사 객체)를 부가하기 위해 추가적인 동물원 사육사들(즉, 초가의 동물원 사육사 클래스들)을 정의하고, 동물원 사육사 클래스 및/또는 객체들을 삭제한다. 이후 명백하게 되는 바와 같이, 각 동물원 사육사 객체는 특정의 동물원 태스크를 수행한다. 따라서, ZAF의 사용자가 추가의 동물원 태스크를 처리하기 위해 동물원 사육사 정의 및 객체를 부가하거나 더 이상 필요없는 정의 또는 객체를 삭제하기를 또한 원할 수 있음은 당연하다. 알 수 있는 바와 같이, 이러한 유연성은 동물원 사육사 메카니즘을 확장 기능으로 설계함으로써 제공된다.

추가/삭제_동물원_사육사() 연산과 마찬가지로, 추가/삭제_동물() 연산은 사용자들과 상호작용하여 추가의 동물원 동물 클래스 및 객체를 정의하고 더 이상 필요없는 클래스 및 객체를 삭제한다. 역시 동물원이 동물들을 추가 및 삭제할 필요가 있음은 아주 당연하다. 추가/삭제_억제_장치() 연산은 새로운 억제 장치 클래스 및 객체를 정의하고 더 이상 필요없는 클래스 및/또는 객체를 삭제한다. 또한, 본 프레임워크 설계자는 동물 메카니즘 및 억제 장치 메카니즘을 확장 기능으로 설계함으로써 이러한 유연성을 제공하는 방식으로 ZAF를 설계하였다.

도 8을 다시 참조하면, 동물원 사육사 클래스 정의는 동물 등록 클래스, 동물 클래스, 억제 장치 등록 클래스 및 억제 장치 클래스와 사용 관계를 갖는다. ZAF의 가치가, ZAF의 소비자들이 동물원 사육사 클래스, 동물 클래스 및 억제 장치 클래스를 커스텀화하고 확장하도록 허용함으로써 향상되므로, 이들 클래스는 확장 기능으로 설계하였다. 그러나 동물 클래스 및 억제 장치 등록 클래스의 동작을 변화시키는 것은 ZAF의 기본적 동작을 분열시키게 된다. 따라서, 이들 클래스는 ZAF의 핵심 기능으로 설계되었다.

도 10은 동물원 사육사 클래스의 클래스 도면이다. 그러나 도 10의 세부사항을 설명하기 전에, 도 10상에 도시된 클래스 정의들이 '클래스 계층(a class hierarchy)'이라고 불리우는 아주 단순한 순서로 정렬됨을 알아둘 필요가 있다. 동물원 사육사 클래스와 마찬가지로, 클래스 계층에서 가장 일반화된/추상적 클래스를 나타내는 클래스는 그 계층의 기본 클래스(base class)라 일컬어진다. 클래스 계층내의 클래스들의 순서는 가장 일반적인 것으로부터 가장 덜 일반적인 것(즉, 일반적인 것으로부터 특수한 것)으로의 순서이다. 보다 덜 일반적 클래스들(예를 들면, 사육자 클래스)은 보다 더 일반적인 클래스 또는 클래스들(즉, 이 경우 동물원 사육사 클래스)로부터 특징을 상속(inherit)받는다고 말한다. 따라서, 사육자, 수의사 및 온도 제어기와 같은 클래스 정의들은 동물원 사육사 클래스의 서브클래스들이 된다. 상속 메카니즘(inheritance mechanisms)은 도 11과 연관된 설명에서 보다 상세히 탐구된다.

도 10에 도시된 바와 같이, 동물원 사육사 클래스 정의는 단일의 연산 정의, 즉, 체크_동물() 연산 정의를 포함한다. 동물원 사육사 클래스 정의는 추상적 클래스(an abstract class)인 것으로서 표시된 것에 또한 주목해야 한다. 추상적 클래스는 그들의 구성요소들로서 생성된 객체들을 갖지 않도록 설계되었으나, 대신에 그들의 서브클래스들에 대한 공통의 인터페이스/프로토콜을 정의하는데 사용된다. 자신의 연산 정의들중 적어도 하나가 순수하게 가상적인 연산 정의(a pure virtual operation definition)일 때 그 클래스를 추상적 클래스라고 한다. 순수하게 가상적인 연산 정의는 그 연산의 서브클래스 정의에 대한 공통의 인터페이스를 정의할 목적으로만 설계된다. 즉, 실질적인 동작의 설계(즉, 데이터 및 연산들)는 서브클래스들 자체에 일임된다. 동물원 사육사 클래스 정의의 경우에, 사육자, 수의사 및 온도 제어기 서브클래스들은 동물원 사육사 클래스에 포함된 순수 가상 체크_동물() 연산 정의의 특정 구현들을 정의한다. 연산이 0으로 세트되면 그 연산은 순수 가상으로서 표시된다.

그러나, 요구 객체들(클라이언트 객체라고 부름)이 서버 객체의 특정 서브클래스를 알 필요없이 서브클래스 요소 객체들(서버 객체라고 부름)을 사용할 수 있도록 순수 가상 연산 정의의 공통 인터페이스는 모든 서브클래스들에 의해 승인되어야 함에 유의해야 한다. 예를 들어, 동물원 관리자 클래스에 의해 정의된 객체는 특정의 동작이 수행될 것이 필요할 때마다, 동물원 사육사 객체와 상호작용한다. 이들 객체에 대한 인터페이스는 추상적 기본 클래스인 동물원 사육사 클래스에서 정의되어 체크_동물() 연산에 대한 서브클래스 정의에 보존되었기 때문에, 동물원 관리자 객체는 서버 객체들중 어느 것의 서브클래스들에 대해 특별한 지식을 가질 필요가 없다. 이것은 동작이 실행되는 방법(즉, 동물원 사육사 서브클래스들의 객체들중 하나에 의해)으로부터 그 동작에 대한 요구(즉, 동물원 관리자 객체의 일부에 대한)를 분리시키는 효과를 갖는다. 추상적 클래스들의 특성을 이용하는 설계들(ZAF 설계와 유사함)은 '다형성(polymorphic)'이라 한다.

다형성은, 어떤 일이 행해지는 방법(소위 구현방법)이 동작이 실제로 수행되는 사실에 의존하는 메카니즘에 영향을 주지 않고 변경 또는 확장되는 것을 허용하기 때문에 OO 프레임워크 설계에 매우 중요하다. 즉, 클라이언트 객체는 단지 어떤 객체들이 어떤 기능들을 수행한다는 것만을 알 필요가 있지, 그들 기능이 실제로 수행되는 방법을 알 필요는 없다. 이것은 적절히 설계된 프레임워크가 미래의 요건을 만족시키기 위해 용이하게 커스텀화 및 확장될 수 있는 한 방법이다.

앞서 설명된 바와 같이, 본 프레임워크 설계자는 동물원 사육사 객체들이 동물 객체 및 억제 장치 객체와 상호작용하여 그들의 태스크를 수행하도록 ZAF를 설계하였다. 도 11은 추상적 클래스인 동물 클래스의 클래스 계층에 대한 클래스 도면이다. 동물 클래스 정의는 동물원 동물들의 특성과 행동을 나타낼 책임을 지므로, 프레임워크 설계자는 이러한 책임을 반영하는 방식으로 추상적 클래스인 동물 클래스를 설계하였다. 도시된 바와 같이, 실시예의 동물 클래스 정의는 급식_회수, 위치 및 온도_범위와 같은 데

이터 정의들과, 획득_온도_범위(), 급식(), 요구_음식(), 요구_수의사_방문() 및 수의사_방문()과 같은 연산 정의들을 포함한다.

이러한 프레임워크에 대한 개괄의 목적으로서는, 각 정의를 상세히 탐구할 필요는 없다. 그러나, 온도_범위 데이터 정의와 획득_온도_범위() 및 급식() 연산 정의들은 잘 생각해낸 프레임워크 설계 선택들의 좋은 예들이다.

급식() 연산 정의는 동물들의 실제 급식을 수행하도록(즉, 도시되지 않은 특수 급식 장치를 통해) 설계된다. 급식() 연산은 순수 가상 연산이다. 또한, 이것은 클래스의 설계가, 요구된 기능을 수행하는 실제 메카니즘이 서브클래스에 의해 정의되게 일임되도록 한 것임을 의미한다. 요구 서브클래스 정의는 서브클래스들의 구성요소들로서 생성된 객체가 특수화된 요구들을 갖는 이와 같은 경우들에서 좋은 설계 선택이다. ZAF에서, 예를 들면, 각 동물 유형은 특수화된 급식 장치에 대한 필요를 가질 것이며, 이것은 일반적인 급식() 연산의 정의를 어렵게 할뿐 아니라 무가치하게 한다.

비교를 위해, 프레임워크 설계자는 획득_온도_범위() 연산을 순수 가상 연산 정의가 아닌 것으로 명시적으로 설계하였다. 이것은 획득_온도_범위() 연산이 일반적으로 디폴트 연산(a default operation)으로 정의되었음을 의미한다. 따라서, 이것은 가상 연산(a virtual operation)으로 간주된다. 디폴트 연산은 서브클래스들에게 일반적 기능을 제공하는데 사용된다. 이들 서브클래스는 단순히 이들 디폴트 연산을 사용할 수 있으며, 또는 재정의함으로써 디폴트 연산을 커스텀화 또는 확장시킬 수 있다. 디폴트 연산을 재정의하는 것은 디폴트 연산을 '오버라이딩(overriding)'한다고 부른다.

포유동물은 동물 클래스의 서브클래스이며, 따라서, 포유동물은 동물 클래스의 모든 특성을 상속받는다. 이 포유동물 클래스는 또한 추상적 클래스로 설계되며, 이것은 또한 포유동물 클래스가 자신의 구성요소로서 생성된 객체를 갖지 않도록 설계되었으며, 대신에 자신의 서브클래스에 대한 공통의 인터페이스를 제공하도록 설계되었음을 의미한다. 포유동물 서브클래스는 또한 육식동물 클래스와 초식동물 클래스라는 서브클래스를 갖는다.

급식() 연산의 정의는 서브클래스들에게 일임되었으므로, 서브클래스인 육식동물과 초식동물은 각각 급식() 연산의 그들 자신의 정의를 갖는다. 또한, 육식을 하는 동물이 초식을 하는 동물과 상이한 요구를 가질 것이므로, 이것은 좋은 설계 선택이다.

온도_범위는 특정 동물의 천연 거주지의 온도와 동일한 온도 범위에 대한 데이터 정의이고, 획득_온도_범위() 연산 정의는 특정 동물에 대한 온도_범위를 검색하여 그것을 요구 클라이언트 객체에게 복귀시키도록 설계된다. 파충류 서브클래스는 온도_범위에 대한 그 자신의 데이터 정의와 획득_온도_범위() 연산에 대한 그 자신의 정의를 포함한다. 이런 식으로, ZAF는 데이터 정의들이 바로 연산 정의들과 마찬가지로 오버라이딩될 수 있음을 나타내도록 설계되었다. 다수의 파충류는 사막같이 밤은 매우 춥고 낮은 매우 뜨거운 조건에서 생활하므로, 파충류 클래스에서 디폴트 온도_범위 정의는 시간과 온도 정보(도 11에는 명시적으로 도시되지 않음)를 포함하도록 오버라이딩되었다. 이것은, 온도 조정이 일일 시간은 물론 억제 장치 자체내의 현재 온도에 기초하여 이루어질 수 있게 함으로써 ZAF로 하여금 파충류 억제 장치를 다른 억제 장치와 다르게 취급하게 하므로, 또 다른 좋은 설계 선택이다.

도 12는 억제 장치 클래스의 하위 레벨 도면을 도시하는 클래스 도면이다. 억제 장치 클래스는 가상 연산 정의인 조정_온도()를 포함한다. 이 조정_온도() 정의는 실제로 (도시되지 않은 가열 및 냉각 메카니즘에 의해) 동물원의 억제 장치내의 온도를 조정하는데 사용되는 인터페이스와 메카니즘을 정의한다.

ZAF 객체들이 상호작용하는 방법

특정 문제에 대한 해를 형성하는 객체들을 설계하는 것 외에, 본 프레임워크 설계자는 또한 개개의 객체들이 상호관련되는 방법을 설계해야 한다. 즉, 객체들은 그들이 설계되었던 방법을 이용하는 방식으로 상호관련되어야 한다. 설명된 바와 같이, 객체의 정의된 연산들이 그 객체에 대해 정의된 데이터에 대해 작용하는 방법을 객체의 동작(object's behaviour)이라고 부른다. 객체들이 독립적인 개체로서 특성화될 수 있는 반면에, 각 객체는 다른 객체와 상호관계할 때 일관성있는 동작을 나타낸다는 것은 또한 매우 중요하다. 일관성있는 동작은, 객체들이 다른 객체의 일관성있는 동작에 의존하여 그들이 스스로 일관성있는 동작을 나타낼 수 있기 때문에 중요하다. 사실, 일관성있는 동작은 너무나 중요해서 객체의 동작은 종종 객체가 다른 객체들과 맺는 '계약(contract)'이라 지칭된다. 객체가 일관성있는 동작을 나타내지 않을 때, 그것은 다른 객체와의 계약을 파기했다고 말하여진다.

한 객체의 동작이 제 2 객체에 의해 제어되는 데이터에 대한 액세스를 필요로 할 때, 그 객체는 제 2 객체의 클라이언트인 것으로 간주된다. 제 2 객체에 의해 제어되는 데이터를 액세스하려면, 클라이언트의 동작들중 하나가 제 2 객체의 동작들중 하나를 호출하여 그 객체에 의해 제어되는 데이터에 대한 액세스를 획득하게 된다. 그리고 나서, 피호출 객체의 동작들중 하나(즉, 이 경우 서버 동작)가 실행되어 피호출 객체에 의해 제어되는 데이터가 액세스 및/또는 조작된다.

도 13은 ZAF의 객체 예들이 동물원을 운영하는데 있어 동물원 직원들을 돕기 위해 상호작용하는 방법을 도시하는 객체 도면이다. 모든 ZAF 객체들의 상호작용의 상세한 분석은 지금 개관을 목적으로는 불필요하다. 그러나 객체가 문제를 해결하기 위해 상호작용하는 방법에 대한 기본적 이해를 얻기 위해 이하의 간단한 제어 흐름을 검토해야 한다.

언급된 바와 같이, 객체는 특정 클래스의 구성요소로 생성된다. 따라서, 동물원 관리자 젤다(Zelda)[객체(706)]는 동물원 관리자 클래스의 구성요소(사실상 유일한 구성요소)인 객체이다. 따라서, 객체 젤다는 ZAF의 전체적인 제어를 책임진다. 동물원 사육사 객체들 모두는 동물원 사육사 등록 객체[객체(700)]에 등록된다. 따라서, 객체 젤다는 동물원 사육사 등록 객체의 리스트_동물원_사육사() 연산을 호출(단계 1)함으로써 현재 동물원 사육사의 리스트를 획득한다. 동물원 사육사 등록 객체는 동물원 사육사 등록 클래스의 구성요소로서 생성되었다. 예시를 목적으로, 이것이 매 5분마다 젤다의 5_분_타이머() 연산의 일부로서 발생한다고 가정하자. 이 때 동물원 사육사 등록 객체는 동물원 사육사들을 리스트함으로써 응답한다(단계 2). 동물원 사육사 리스트는 온도 체커인 티

나(Tina)[객체(714)], 수의사인 빈스(Vince)[객체(740] 및 동물 사육자인 프레드(Fred)[객체(752)]를 포함한다. 각각의 동물원 사육사는 동물원 사육사 클래스의 구성요소로서 생성되었다. 보다 구체적으로, 온도 체커인 티나 객체, 수의사인 빈스 객체 및 사육자인 프레드 객체는 제각기 온도 제어기 서브클래스, 수의사 서브클래스 및 사육자 서브클래스의 구성요소들이다. 온도() 연산은 다형성이다. 즉, 티나 객체의 체크_동물() 연산은 각각의 조정_온도() 연산이 자신의 태스크를 어떻게 수행하는지에 대한 특별한 지식을 요구하지 않는다. 체크_동물() 연산은 단지 인터페이스를 지속하고 조정_온도() 연산을 호출해야 할 뿐이다. 그 후, 적당한 방법으로 그들의 태스크를 수행하는 것은 개개의 조정_온도 연산에 달려 있다.

이 시점에서, 본 ZAF 메카니즘이, 초보자들로 하여금 본 발명의 잇점 및 장점을 가장 잘 이해할 수 있도록 몇가지 기본적인 프레임워크 개념들을 이해하는 것을 돕기 위해 제공된 매우 간략화된 프레임워크 메카니즘임을 알아 둘 필요가 있다. 이들 잇점 및 장점은 본 상세한 설명중 이하의 부분들을 참조함으로써 더욱 명백하게 된다.

이제 도 14a, 14b 및 15를 보다 구체적으로 참조하면, 도 14a 및 14b는 앞서 설명된 환경에서 본 발명에 따른 프레임워크의 예로서 이해될 수 있으며, 도 15는 그 프레임워크내에서 사용된 특정 객체들의 예이다. 도 14b는 고급 블록도이며, 하급 상세도는 도 14a에 도시된다. 도 14a 및 도 14b의 프레임워크는 몇가지 구성요소들, 즉, 웹 브라우저, 매핑 블록, 서비스 층 및 오퍼레이팅 시스템을 갖는다.

본 실시예에 따르면, 매핑 블록은 인터넷상의 임의의 브라우저를 이용하는 임의의 사용자에게 시스템 관리 프로토콜의 제어 및 이벤트 감시를 제공한다. 이 매핑 블록은 프로토콜 서비스에 대해 내부의 데이터를 브라우저상에서 보기 위해 HTML로 변환하며, HTML로서 사용자의 입력을 프로토콜 서비스의 제어로 변환한다.

본 발명의 사용자는 오늘날 이용가능한 어떤 것일 수도 있는 웹 브라우저로부터 시스템을 감시 및 제어한다.

매핑 블록은 인터넷을 통해 웹 브라우저에게 정적 형식(static forms)(HTML에서) 또는 동적 애플릿(dynamic applets)(예를 들면, 자바(Java)에서)을 제공한다. 이들 형식 또는 애플릿은 시스템 관리 프로토콜로부터의 정보의 감시 및/또는 시스템 관리 프로토콜내의 이벤트 발생에 대한 입력 수단을 제공한다.

브라우저로의 사용자 입력은 그가 모니터하기를 원하는 것과 제어하기를 원하는 방법을 표시한다. 사용자의 입력은 형식 또는 애플릿으로 제공되는 인스트럭션들내에서 브라우저에 의해 매핑 블록으로 되전송된다. 매핑 블록과 브라우저간의 데이터는 오늘날 통상적인 바와 같이 HTTP 프로토콜을 사용하여 인터넷을 통해 HTML과 같은 표준 언어로 전송된다.

매핑 블록은 시스템 관리 프로토콜의 서비스 층과 직접 통신한다. 사실, 매핑 블록은 서비스에 대해 GUI가 된다. 서버는 인터넷을 통해 수신된 HTML을 서버 층에 제공된 커맨드로 변환하며, 서버 층은 오퍼레이팅 시스템을 통해 요구를 실행한다.

본 발명은 DMI, SNMP, CMIP 및 CMOL 시스템 관리 프로토콜에 대해 인터넷을 통한 플랫폼-독립 브라우저를 허용한다.

매핑 블록은 웹 브라우저로부터의 HTTP 요구를 서비스하고, 그들을 서비스 층에 특정한 호출들로 변환하는 HTTP 서버로서 설계될 수 있다. 서비스 층으로부터 복귀된 데이터는 HTML로 변환되어 블라우저에게로 복귀된다. 이것은 본질적으로 플랫폼-독립 웹 브라우저로부터 DMI, SNMP, CMIP, CMOL 또는 어떤 다른 시스템 관리 프로토콜의 관리를 용이하게 한다.

다른 실시예로서, 도 14a 및 14b의 프레임워크는 다음과 같은 구성요소들, 즉, HTTP 리퀘스터, 매핑 블록, 트랜스포트 층, HTTP 서버를 포함한다.

본 발명의 이 실시예는 매핑 블록을 통해 HTTP를 트랜스포트 층에 의해 사용되는 프로토콜 독립 커맨드로 변환하는 프레임워크를 제공한다. HTTP 리퀘스터와 매핑 블록간의 통신은 두가지 유형들중 하나일 수 있다. 매핑 블록은 TCP/IP로부터 HTTP 요구를 수신하여 수신된 요구를 NetBIOS, IPX 또는 직렬 프로토콜로 트랜스포트 층을 통해 경로배정하는 게이트웨이 HTTP 서버일 수 있다. 이러한 구성은 일반 웹 브라우저를 사용하여 TCP/IP에 직접 접속되지 않은 HTTP 서버에 도달할 수 있게 한다. TCP/IP 접속을 전혀 갖지 않는 이들 컴퓨터 장비들의 경우, HTTP 리퀘스터 또는 웹 브라우저는 매핑 블록과 직접 통합되어야 한다. HTTP 서버는 항상 그의 매핑 블록과 통합된다.

본 발명은 브라우저의 후미를 매핑 블록을 포함하도록 재작성함으로써, 또는, 브라우저에 의해 송신된 HTTP 리퀘스터에 대한 게이트웨이로서 서비스하도록 매핑 블록을 개발함으로써 구현될 수 있다.

또 다른 실시예로서, 도 14a 및 14b의 프레임워크는 다음 구성요소들, 즉, 웹 브라우저, 매핑 블록, 트랜스포트 층 및 시스템 관리 베이스를 포함한다.

본 실시예에 따르면, 매핑 블록은 인터넷상의 임의의 브라우저를 이용하는 임의의 사용자에게 시스템 관리 모니터링 및 제어를 제공할 수 있게 한다. 매핑 블록은 시스템 관리 소프트웨어에 대해 내부의 데이터를 브라우저상에서 보기 위한 HTML로 변환하며, HTML로서 사용자의 입력을 시스템 관리 소프트웨어에 대한 제어로 변환한다. 여기서 프레임워크는 관리 프레임워크라 불릴 것이다. 웹 브라우저를 제외하고, 프레임워크의 구성요소들은 서버 컴퓨터 시스템상에서 실행될 수도 있다. 본 발명의 사용자는 임의의 일반 웹 브라우저로부터의 관리하에 시스템을 모니터하고 제어한다.

매핑 블록은 인터넷을 통해 웹 브라우저에게 정적 형식(HTML에서) 또는 동적 애플릿(예를 들면, 자바에서)을 제공한다. 이들 형식 또는 애플릿은 트랜스포트 층(Transport layer)을 통해 시스템 관리 베이스를 액세스함으로써, 관리하에 있는 시스템의 제어를 위한 입력 수단을 제공하고, 또는 관리하에 있는 시스템으로부터의 정보를 모니터링한다. 브라워저로의 사용자의 입력은 그가 모니터하기를 원하는 것 또는 제어하고자 하는 방법을 표시한다. 사용자의 입력은 형식 또는 애플릿에 의해 제공된 인스트럭션들

내에서 브라우저에 의해 매핑 블록으로 되전송된다.

매핑 블록과 웹 브라우저간에 교환되는 데이터는 오늘날 통상적인 바와 같이 HTTP 프로토콜을 사용하여 인터넷을 통해 HTML과 같은 표준 언어로 전송된다.

매핑 블록은 트랜스포트 층 및 시스템 관리 베이스를 통해 관리하에 있는 시스템과 통신한다. 사실, 매핑 블록은 GUI-기반 범예에서 GUI가 된다. 매핑 블록은 인터넷을 통해 수신된 HTML를 트랜스포트 층에 제공되는 프로토콜-독립 커맨드로 변환한다.

트랜스포트 층은 NetBIOS, TCP/IP, IPX 또는 직렬 링크를 통해 시스템들간에 이들 커맨드를 전송한다.

시스템 관리 베이스는 트랜스포트 층으로부터 커맨드를 수신한다. 그리고 나서, 시스템 관리 베이스는 사용자의 인스트럭션들을 실행하거나 컴퓨터 시스템 자체로부터 바람직한 모니터링을 요구한다.

본 발명이 웹을 통해 제어할 수 있는, 시스템 관리 베이스에서 이용가능한 서비스는 다음과 같다.

시스템 정보 툴:이것은 시스템의 하드웨어 및 소프트웨어 구성에 대한 포괄적인 정보를 제공한다.

시스템 모니터 서비스:이것은 시스템에서 측정가능한 양의 모니터링을 허용한다. 이들 양은 관측되어 수집된 후 데이터베이스로 수송된다. 또한, 이들 양에 대한 임계치를 설정하여 경보를 발생할 수도 있다.

시스템 파티션 액세스 서비스:이것은 시스템의 시스템 파티션에 대한 관측 및 제어를 허용한다.

경보 관리자 서비스:이것은 경보 수집기로부터의 프로화일에 의한, 경보 기록 조사, 경보 발생시 취해질 동작 설정, 경보 클래스를 캡슐화하는 프로화일(profiles) 정의 및 경보 요구를 허용한다.

보안 관리자 서비스:이것은 서비스 레벨에 대한 액세스를 제한하기 위해 사용자 및 패스워드를 정의할 수 있게 한다. 또한, 이것은 안전한 액세스에 대해 경보를 설정할 수 있게 한다.

ECC 메모리 서비스:이것은 ECC 메모리 에러 및 고장에 대해 감시하여 경보를 발생할 수 있게 한다.

예측 고장 분석 서비스:이것은 디스크 에러 및 고장에 대해 모니터링하여 경보를 발생할 수 있게 한다.

시스템 프로화일 서비스:이것은 시스템에 대한 중요한 어카운팅(accounting) 및 재고 정보를 정의 및 입력할 수 있게 한다.

중요 파일 모니터:이것은 파일들에 있어서의 변화를 검출할 수 있도록 파일들을 감시하여 경보를 발생할 수 있게 한다.

넷휘니티 직렬 제어(NetFinity Serial Control):이것은 내부 프로토콜에 의한 통신을 위해 직렬 라인을 정의 및 설정할 수 있게 한다.

RAID 관리:이것은 RAID 드라이브 제어와 그의 에러 및 고장에 대한 감시 및 경보를 허용한다.

원격 세션 서비스(remote session service):이것은 원격 시스템에 대해 터미널 쉘 능력(terminal shell capability)을 허용한다.

파일 전송 서비스:이것은 근거리 시스템과 원격 시스템간에 파일 또는 디렉토리의 전송을 허용한다.

스크린 뷰 서비스:이것은 실시간으로 시스템 입력 및 출력을 캡처(capture)함은 물론 근거리 또는 원격 시스템에 대해 가시 스크린을 캡처하여 볼 수 있게 한다.

원격 시스템 관리자 서비스:이것은 시스템 그룹의 정의 및 유지보수를 가능하게 한다. 또한, 이것은 시스템의 상태 또는 존재에 대한 경보 발생을 허용한다.

서버보호 지원 서비스:이것은 서버 시스템상에서 하드웨어 모니터의 상태(전원 공급 장치, 온도 등)에 대해 감시하여 경보할 수 있게 한다.

이벤트 스케줄러 서비스:이것은 스케줄링된 서비스 또는 커맨드를 정의 및 활성화시킬 수 있게 한다.

파워-온 에러 검출:이것은 시스템이 파워-온되어 부팅되는 사이에 발생하는 에러들을 경보하고 캡처할 수 있게 한다.

데이터베이스 지원:이것은 서비스 데이터의 데이터베이스 수출을 위해 파라미터를 설정 및 배열할 수 있게 한다.

소프트웨어 재고:이것은 시스템상에 인스툴된 소프트웨어를 정의 및 감시할 수 있게 한다.

DMI 브라우저:이것은 DMI 이벤트들을 관측하고 발생할 수 있게 한다.

보고:이것은 다양한 모니터링 서비스로부터 생성된 보고서를 설계하고, 발생하며 볼 수 있게 한다. 이러한 보고는 다양한 서비스에 의해 전달된 실시간 데이터를 보완하는 이력 데이터(historical data)의 정적인 베이스로서 작용한다. 또한, 이들 보고는 시스템 관리 임무로부터 멀리 떨어진 사람에 의해 사용될 것이다. 플랫폼-독립 웹 인터페이스를 가지면 이러한 목적에 특히 유용하다.

HTTP 서버는 인터넷에 접속된 포트상의 HTML을 수신하는 매핑 층으로써 작성되었다. 이 서버는 대개, IBM이 제공하는 넷휘니티(넷휘니티(NetFinity)는 IBM사의 등록상표임)로서 알려진 것과 같은 공지의 시스템 관리 프로그램에 이미 제공된 서비스의 각각에 대해 그래픽 사용자 인터페이스를 재작성한다. 중요한 차이는, 이 서버가 윈도우 시스템으로 상호작용하는 대신에 HTML로 상호작용한다는 것이다.

더욱이, 많은 강력한 시스템 관리 능력은 웹/브라우저 예에 대해서는 당연하며, 전용 GUI에서는 그보다 어렵다.

매핑 블록내에서 발견되는 객체들이 도 15에 더욱 충분히 예시되며, 이 도면은 매핑 블록의 내부를 도시한다. 도 14에서 알 수 있는 바와 같이, 매핑 블록은 웹 브라우저와 애플리케이션 사이에 상주하며, 인터넷을 통해 관찰되고 제어된다. 매핑 블록은 두 개의 객체, 즉, HTTP 객체와 HTML 객체로 구성된다.

HTTP 객체는

- 인터넷 통신을 개시 및 유지하며,

- 인터넷을 통해 HTTP 요구를 수신하며,

- 요구를 형성하는 URL 및 임의의 다른 데이터를 파싱하며,

- URL, 입력 데이터 및 브라우저 ID를 HTML 객체에게로 전송하며,

- HTML 객체로부터 HTML 소스를 수신하여 그것을 요구 브라우저에게 복귀시키는 역할을 한다.

도 15에 도시된 바와 같이, HTTP 객체는 브라우저 유형에 의해 서브클래스로 더욱 분류될 수 있다. 이와 같이 하면, HTTP 객체는 HTML 필터로서 동작하게 되며, HTML 객체에 의해 복귀된 HTML 소스를 특정의 브라우저에 의해 잘 사용될 수 있는 소스로 회유(massaging)시키게 된다. 3개의 '브라우저' 예가 도 15에 제공되어 있다. 넷스케이프로서 알려진 상업적으로 이용가능한 제품과 같은 전기능형 브라우저를 서비스하는 경우, HTTP 객체는 HTML 객체로부터 브라우저로 전송되는 HTML을 전혀 변화시키지 않을 수도 있다. 그러나 보다 덜 능력이 있는 브라우저, 예를 들면, 아주 정확히 HTML 1.0을 처리하는 브라우저를 서비스하는 경우, 보다 고급의 서비스를 요구하는 HTML 소스 구성요소들은 이러한 환경에서 의미있게 디스플레이될 수 있는 소스로 필터링된다. 더욱이, HTTP 객체는 커맨드 라인(a command line) '브라우저'를 처리하도록 서브클래스화될 수 있다. 이 경우, HTML은 터미널 스크린상에 의미있게 디스플레이되기 위해 모든 마크업을 제거하고 적절한 공간을 삽입하여 기본적으로 변경되어야 한다.

HTML 객체는

- URL 및 다른 요구 데이터를 수신하며,

- 이들 요구를 이해하여 애플리케이션의 인터페이스 표시로 그들을 형성하며,

- 애플리케이션으로부터 응답을 수신하고,

- 그들 응답을 HTML로 변환하여 그 HTML을 HTTP 객체로 전송하는

역할을 한다.

도 15에 도시된 바와 같이, HTML 객체는 또한 브라우저 유형으로 더욱 서브클래스화될 수 있다. HTTP 객체의 경우에서처럼, 이러한 서브클래스화의 목적은 특정 유형의 브라우저에 의해 가장 잘 이해되는 HTML을 생성하는 것이다. 그러나 HTTP 객체는 단순히 입력으로서 HTML을 취하고, 단지 HTML/HTML 또는 HTML/ASCII 필터로서만 작용하여 입력 HTML로 표시된 실제 데이터에 전혀 영향을 주지 않는다. 대조적으로, HTML 객체의 단계에서, 애플리케이션과의 인터페이스 설계자는 또한 데이터가 진정 무엇을 나타내는지를 알고 있다. 따라서, 설계자는 이 단계에서 브라우저의 선택에 대해 반작용하여, 데이터를 그 브라우저의 ID에 의존하는 HTML 또는 ASCII로 포맷할 수 있다. 간단한 예로 브라우저에 의한 그래프의 요구가 있을 수 있다. 브라우저가 넷스케이프 브라우저인 경우, HTML 객체는 GIF로서 전송된 그래프를 발생할 수 있고, 다른 한편, '브라우저'가 커맨드 라인인 경우, 브라우저는 그래프의 ASCII 근사치를 발생하게 된다.

본 발명의 일실시예에 따르면 두 개의 객체가 관련된다.

제 1 객체는 HTTP 객체이다. 이 객체는 HTTP 접속을 캡슐화한다. 그의 구성요소는 TCP/IP 소켓과 요구된 URL이며, 그의 메쏘드는 경로 획득(get the path) 메쏘드, 서비스 획득(get the service) 메쏘드, 사용자 획득(get the user) 메쏘드, 패스워드 획득(get the password) 메쏘드, 능력 획득(get the capability) 메쏘드 및 특정 인수 획득(get the particular arguments) 메쏘드이다.

제 2 객체는 HTML 객체이다. 이 객체는 HTML/애플리케이션 인터페이스를 캡슐화한다.

그의 구성요소는 HTML 데이터이며, 그의 메쏘드는 텍스트 부가(adding text) 메쏘드, 마크업 부가(adding of markup) 메쏘드 및 브라우저-특정 마크업 부가(adding of browser-specific markup) 메쏘드이다.

이상 본 발명의 실시예가 HTTP 객체 및 HTML 객체에 대한 한 세트의 클래스 라이브러리로서 구현되었다. HTML 객체는 특정 브라우저에 특정된 마크업 특징을 제공하도록 서브클래스화된다. 본 발명은 웹 서버에 의해 발생된 HTML을, 예를 들면, MVS 또는 VT100 브라우저와 같은 특정 브라우저에 적합시키는데 사용된다.

(57) 청구의 범위

청구항 1

클라이언트 컴퓨터 장치(a client computer device)가 서버 컴퓨터 장치(a server computer device)에 위치된 정보를 액세스할 수 있게 하는 정보 운용 시스템(an information handling system)-상기 클라이언트 컴퓨터 장치는 하이퍼텍스트 전송 프로토콜(HyperText Transfer Protocol;HTTP)을 사용하여 상기 서버 컴퓨터 장치와 통신하는 수단과, 하이퍼텍스트 마크업 언어(HyperText Markup Language;HTML)를 사용하여 상기 서버 컴퓨터 장치로부터 수신된 정보를 상기 클라이언트 컴퓨터 장치의 사용자에게 제공하는 수단을 포함함-에 있어서,

① 상기 클라이언트 컴퓨터 장치 및 서버 컴퓨터 장치들 사이에 위치되며, HTTP 이외의 네트워크 전송

프로토콜(a network transmission protocol)과 HTML 이외의 네트워크 데이터 포맷(a network data format)을 사용하는 네트워크와,

② 상기 네트워크를 통해, 상기 클라이언트 컴퓨터 장치 및 상기 서버 컴퓨터 장치로부터 송신되거나 그에 의해 수신된 HTML/HTTP 정보를 상기 네트워크 전송 프로토콜 및 상기 네트워크 데이터 포맷으로 변환하는 전송 프로토콜 변환 수단

을 포함하는 정보 운용 시스템.

청구항 2

제 1 항에 있어서,

상기 전송 프로토콜 변환 수단은 객체 지향형 컴퓨터 소프트웨어(object oriented computer software)를 실행하는 정보 운용 시스템.

청구항 3

제 1 항에 있어서,

상기 클라이언트 컴퓨터 장치는 인터넷(Internet)에 접속되며, 상기 네트워크는 인터넷과 상기 서버 컴퓨터 장치 사이에 위치되는 정보 운용 시스템.

청구항 4

제 1 항에 있어서,

상기 클라이언트 컴퓨터 장치의 정보를 제공하는 수단은 특정 유형의 그래픽 사용자 인터페이스(a specific type of graphical user interface)를 사용하고,

상기 클라이언트 컴퓨터 장치는 인터넷에 접속되고 상기 네트워크는 인터넷과 상기 서버 컴퓨터 장치 사이에 위치되며,

상기 전송 프로토콜 변환 수단은 객체 지향형 컴퓨터 소프트웨어를 실행하되, 상기 소프트웨어는

㉑ 상기 클라이언트 컴퓨터 장치로부터의 HTML 요구 데이터를 수신하여 상기 수신된 HTML 요구 데이터를 네트워크 데이터 포맷으로 변환하는 HTML 객체와,

㉒ 상기 클라이언트 컴퓨터 장치와 상기 서버 컴퓨터 장치사이의 인터넷 통신을 개시 및 유지하고, 상기 클라이언트 컴퓨터 장치로부터 수신된 HTML 요구 데이터를 파싱(parse)하며, 상기 클라이언트 컴퓨터 장치에 의해 사용된 특정의 그래픽 사용자 인터페이스의 ID(identity)를 검출하고, 상기 요구 데이터 및 상기 검출된 그래픽 사용자 인터페이스 ID를 상기 HTML 객체로 전송하며, 상기 네트워크 및 인터넷을 통해 상기 HTML 객체와 상기 클라이언트 컴퓨터 장치간에 HTML 데이터를 전송하는 HTTP 객체를 포함하는

정보 운용 시스템.

청구항 5

제 3 항에 있어서,

상기 서버 컴퓨터 장치는 시스템 관리 소프트웨어 프로그램(a system management software program)을 실행하는 시스템 관리 수단을 포함하고, 상기 네트워크 프로토콜은 시스템 관리 프로토콜이며, 상기 클라이언트 컴퓨터 장치로부터의 HTML/HTTP 데이터는 상기 시스템 관리 프로그램을 제어하기 위한 커맨드(command)를 포함하는 정보 운용 시스템.

청구항 6

제 5 항에 있어서,

상기 시스템 관리 소프트웨어는 상기 클라이언트 컴퓨터 장치 및 상기 서버 컴퓨터 장치의 하드웨어 및 소프트웨어 구성에 관한 데이터를 처리하는 정보 운용 시스템.

청구항 7

정보 운용 시스템용 프로그램 요소(a program element)에 있어서,

상기 정보 운용 시스템 상에서 실행되었을 때, 데이터를 (a) HTML 및 HTTP 와 (b) 이 HTML 및 HTTP 이외의 커맨드 및 프로토콜에 대해 매핑하고, HTML 객체 및 HTTP 객체를 갖는 매핑 블록(a mapping block)을 포함하되,

상기 HTML 객체는 URL 및 다른 요구 데이터를 수신하며, 수신된 요구를 HTML 및 HTTP 이외의 내부 커맨드 및 프로토콜로 형성하고, HTML 및 HTTP 이외의 내부 커맨드 및 프로토콜로 응답(replies)을 수신하며, 수신된 응답을 HTML로 변환하고, 이러한 HTML을 상기 HTTP 객체에게로 전송하며,

상기 HTTP 객체는 인터넷 통신을 개시 및 유지하며, 인터넷을 통해 HTTP 요구를 수신하고, 수신된 요구를 형성하는 URL 및 연관된 데이터를 파싱하며, 상기 URL 및 연관된 데이터와 브라우저 프로그램 ID를 상기 HTML 객체에게로 전송하고, 상기 HTML 객체로부터 HTML 소스를 수신하며, 상기 HTML 소스를 HTTP 응답으로서 인터넷으로 복귀시키는

정보 운용 시스템용 프로그램 요소.

청구항 8

제 7 항에 있어서,

상기 HTML 객체는 그의 구성요소로서 TCP/IP 소켓(socket)과 요구된 URL을 가지며, 메쏘드로서 경로 획득(get the path) 메쏘드, 서비스 획득(get the service) 메쏘드, 사용자 획득(get the user) 메쏘드, 패스워드 획득(get the password) 메쏘드, 능력 획득(get the capability) 메쏘드 및 특정 인수 획득(get the particular arguments) 메쏘드를 갖는 정보 운용 시스템용 프로그램 요소.

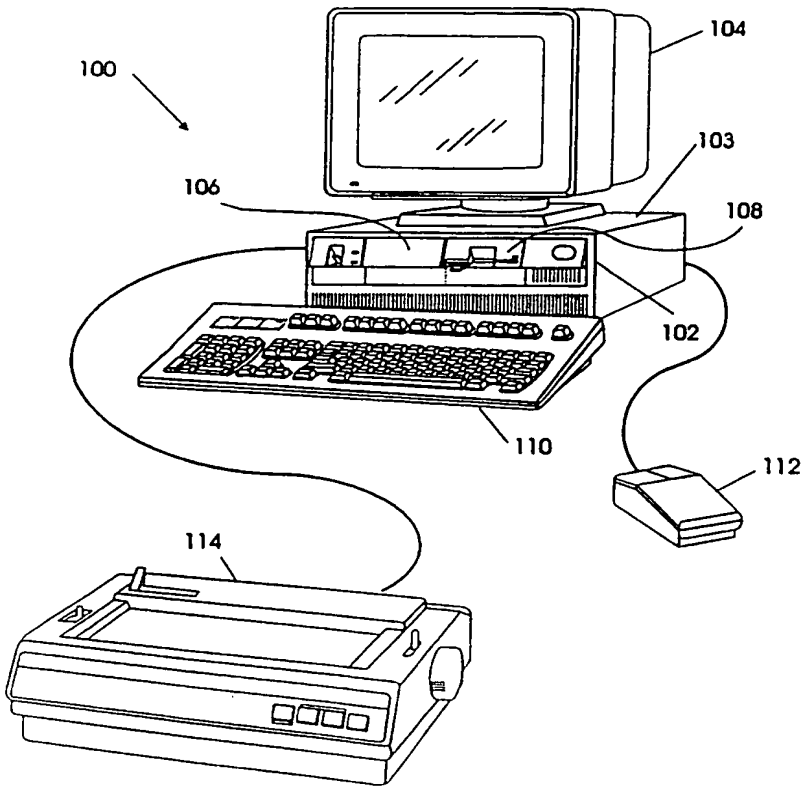
청구항 9

제 7 항에 있어서,

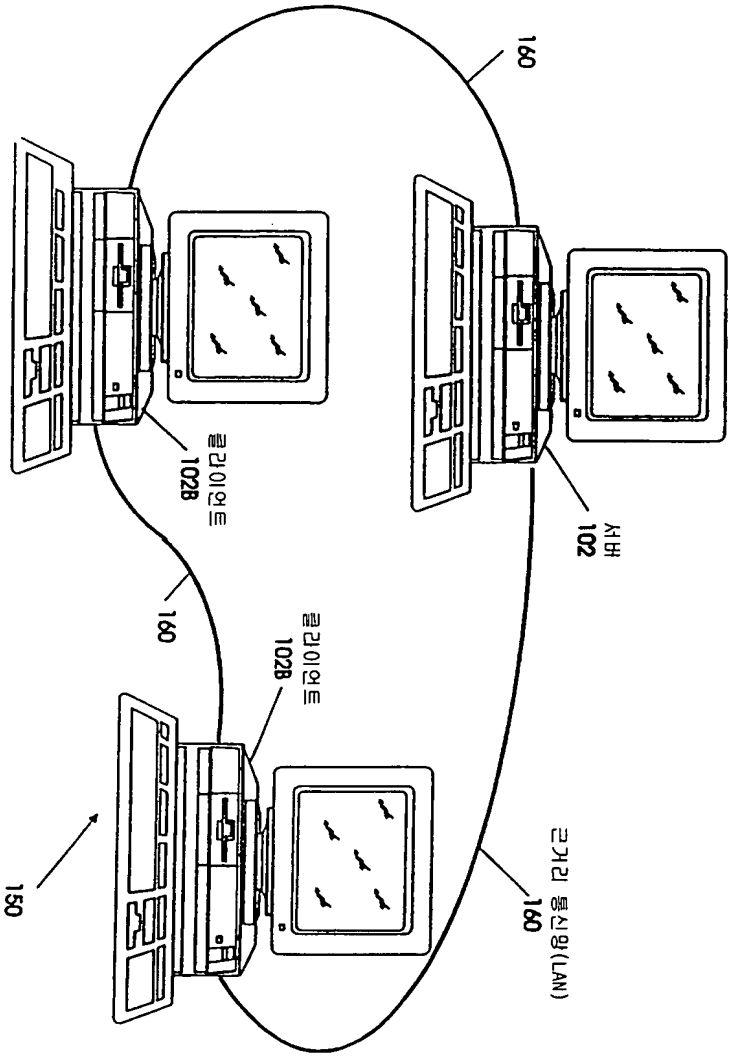
상기 HTTP 객체는 그의 구성요소로서 HTML 데이터를 가지며, 메쏘드로서 텍스트 부가(adding text) 메쏘드, 마크업 부가(adding of markup) 메쏘드 및 브라우저-특정 마크업 부가(adding of browser-specific markup) 메쏘드를 갖는 정보 운용 시스템용 프로그램 요소.

도면

도면1a



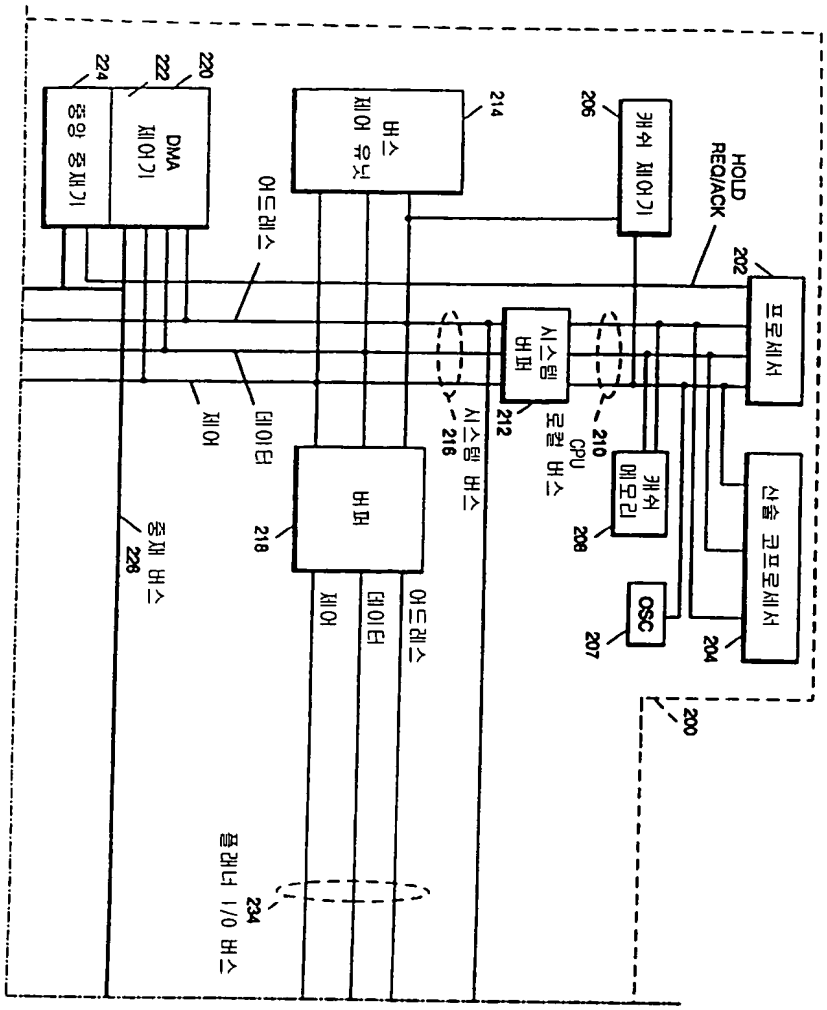
도면1b



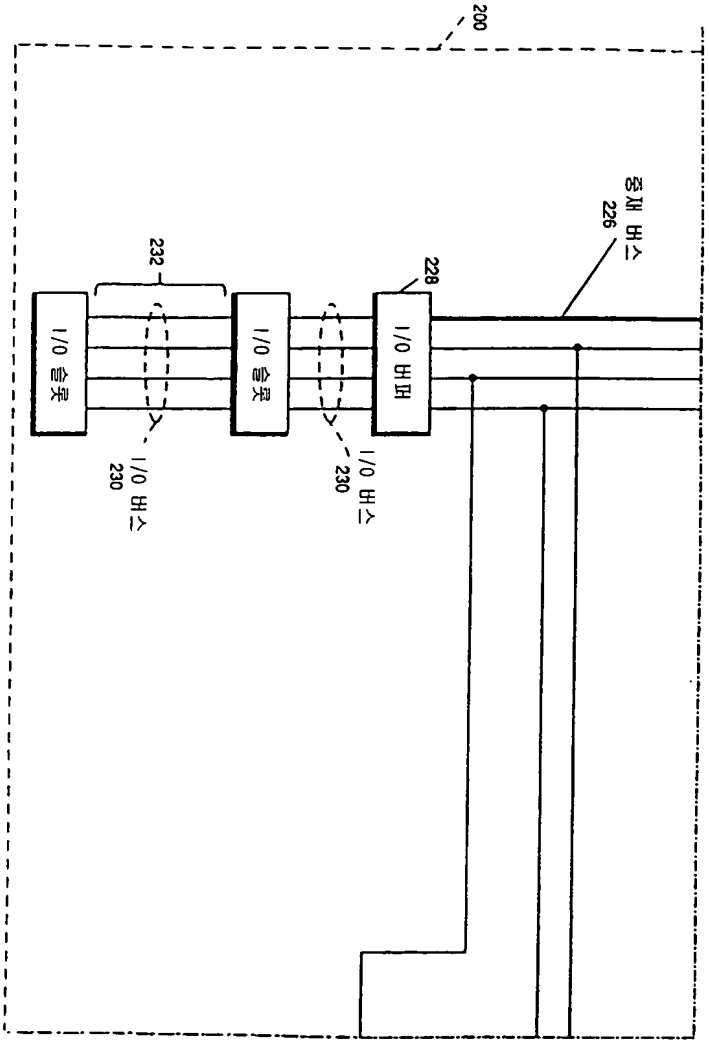
도면2

도 2a	도 2c
도 2b	도 2d

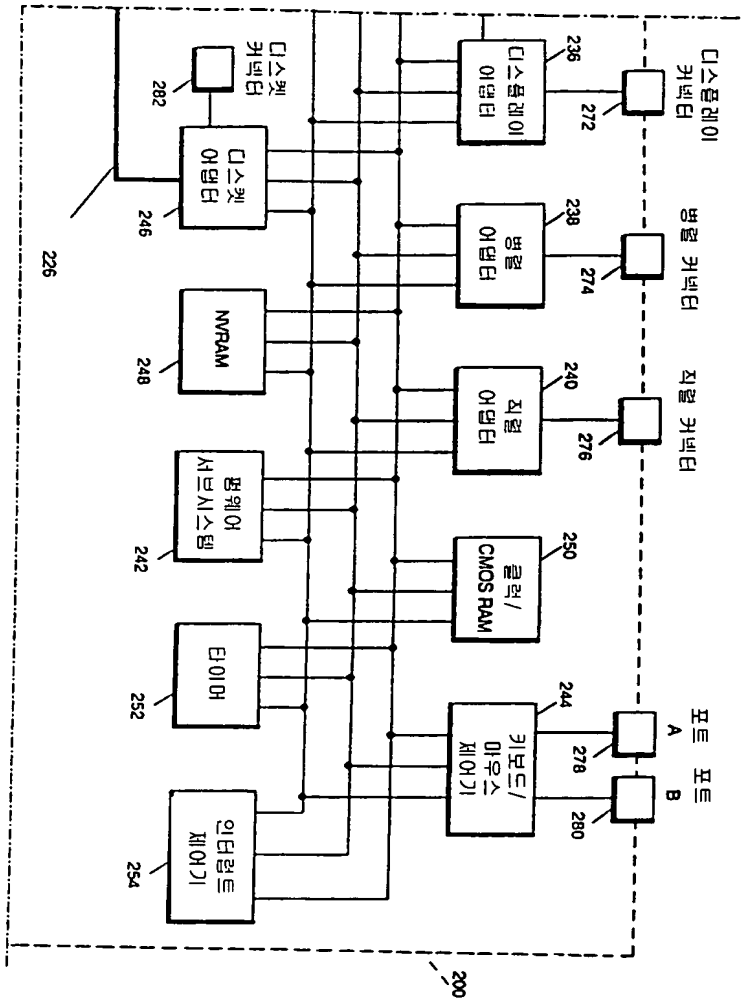
도면2a


220
222
224
DMA 제어기
중앙 중재기
214
버스 제어 유닛
206
캐쉬 제어기
HOLD REQ/ACK
202
프로세서
어드레스
시스템 버퍼
212
210
CPU 로컬 버스
캐쉬 메모리
208
산술 코프로세서
204
OSC
207
200
제어
데이터
시스템 버스
216
버퍼
218
중재 버스
228
제어
데이터
어드레스
플래너 I/O 버스
234

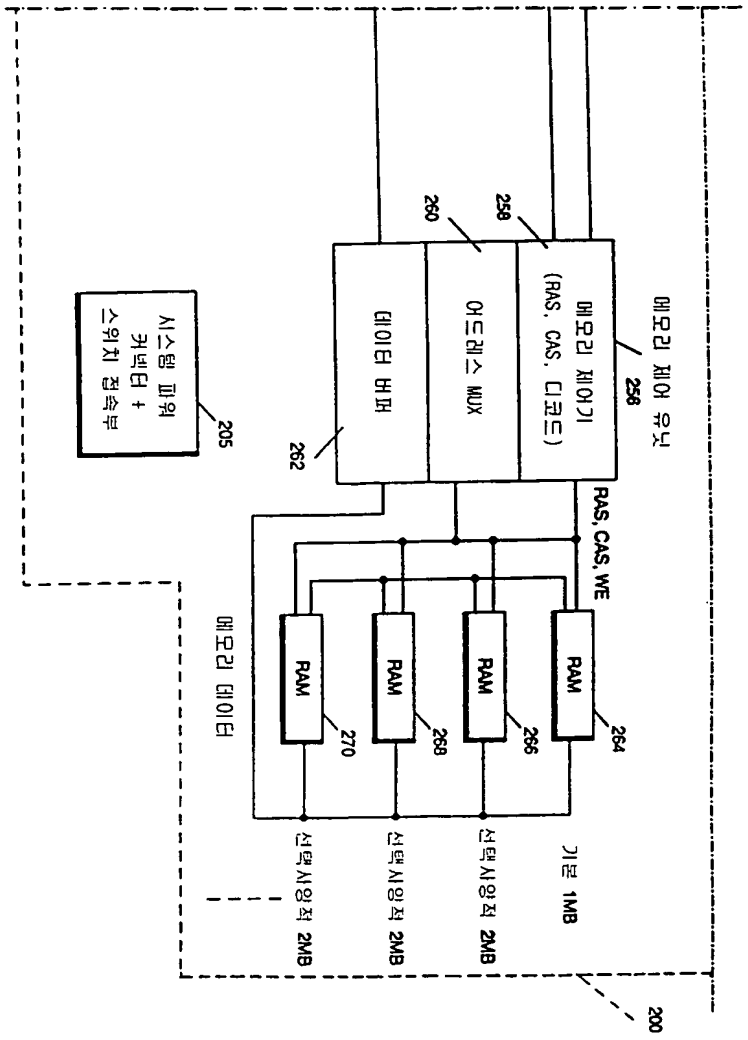
도면2b



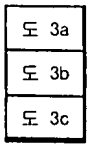
도면2c


디스플레이 커넥터
병렬 커넥터
직렬 커넥터
포트 A
포트 B
디스켓 커넥터
디스플레이 어댑터
병렬 어댑터
직렬 어댑터
키보드/마우스 제어기
클럭/CMOS RAM
디스켓 어댑터
NVRAM
펌웨어 서브시스템
타이머
인터럽트 제어기
236
272
238
274
240
276
244
278
280
250
246
248
242
252
254
282
226
200

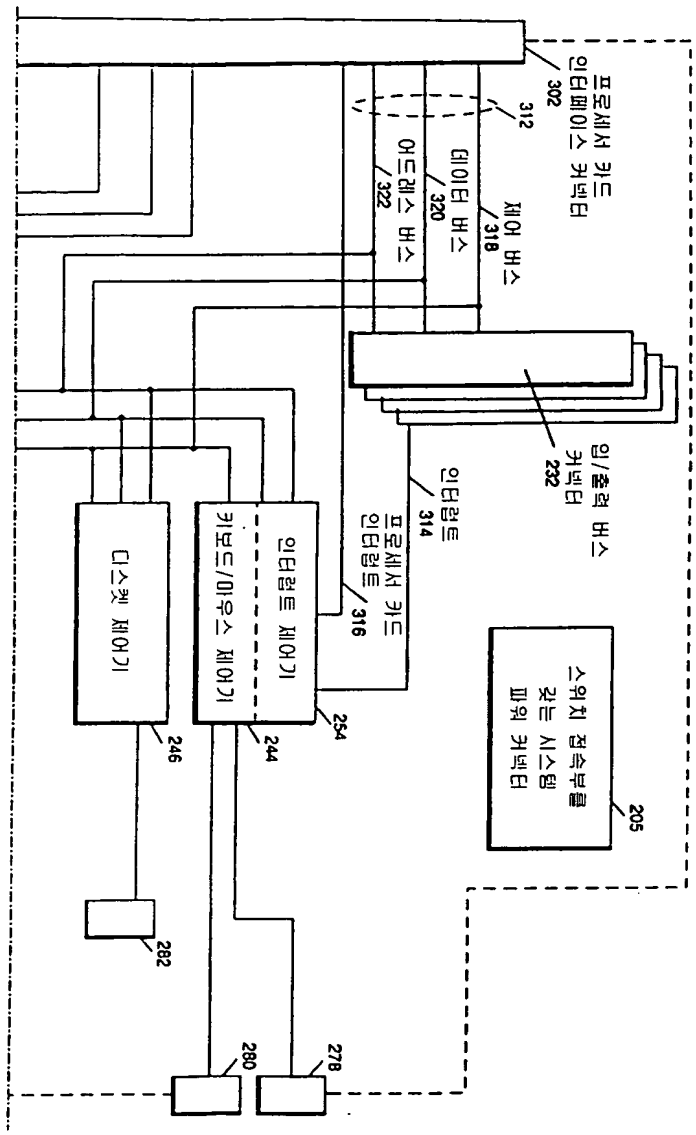
도면2d


메모리 제어 유닛
메모리 제어기 (RAS, CAS, 디코드)
256
258
어드레스 MUX
260
데이터 버퍼
262
RAS, CAS, WE
RAM
264
RAM
266
RAM
268
RAM
270
메모리 데이터
선택사항적 2MB
선택사항적 2MB
선택사항적 2MB
기본 1MB
시스템 파워 커넥터 + 스위치 접속부
205
200

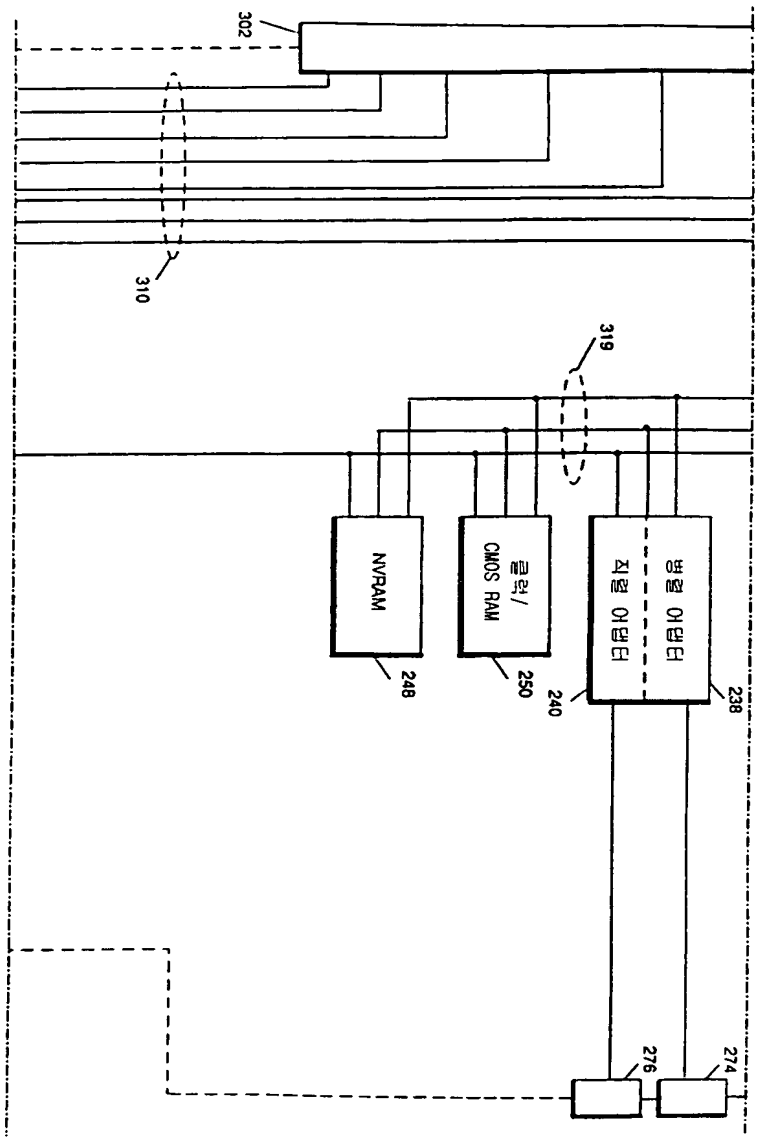
도면3


도 3a
도 3b
도 3c

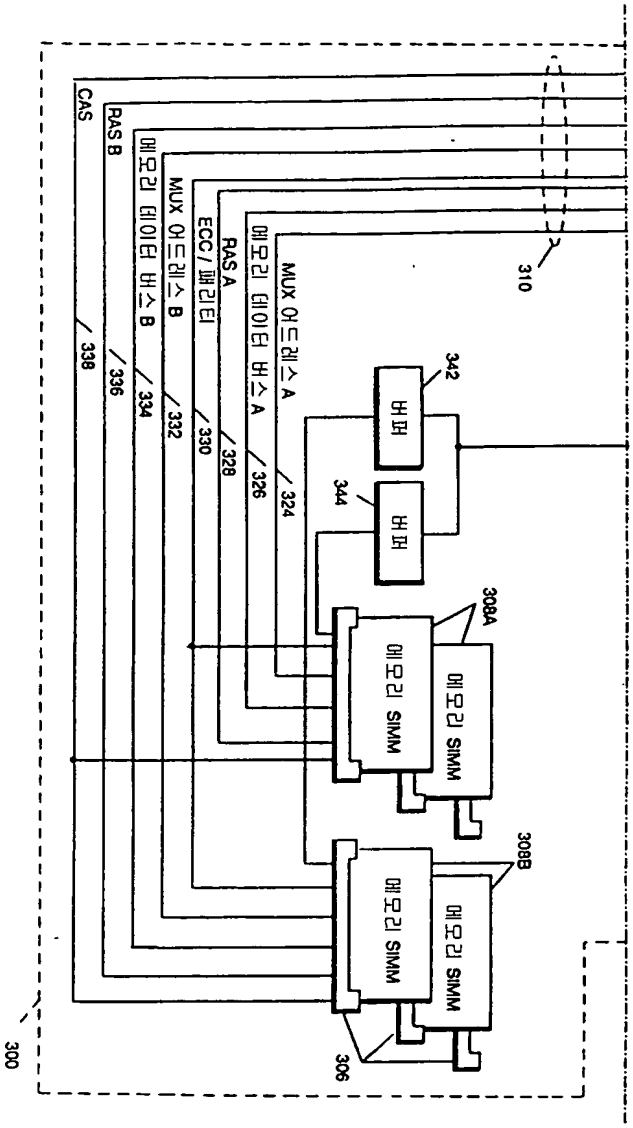
도면3a


프로세서 카드
인터페이스 커넥터
302
312
데이터 버스
320
제어 버스
318
어드레스 버스
322
입/출력 버스
커넥터
232
인터럽트
314
프로세서 카드
인터럽트
316
스위치 접속부를
갖는 시스템
파워 커넥터
205
디스켓 제어기
키보드/마우스 제어기
인터럽트 제어기
254
246
244
282
280
278

도면3b


302
310
319
NVRAM
248
클럭/
CMOS RAM
250
직렬 어댑터
240
병렬 어댑터
238
276
274

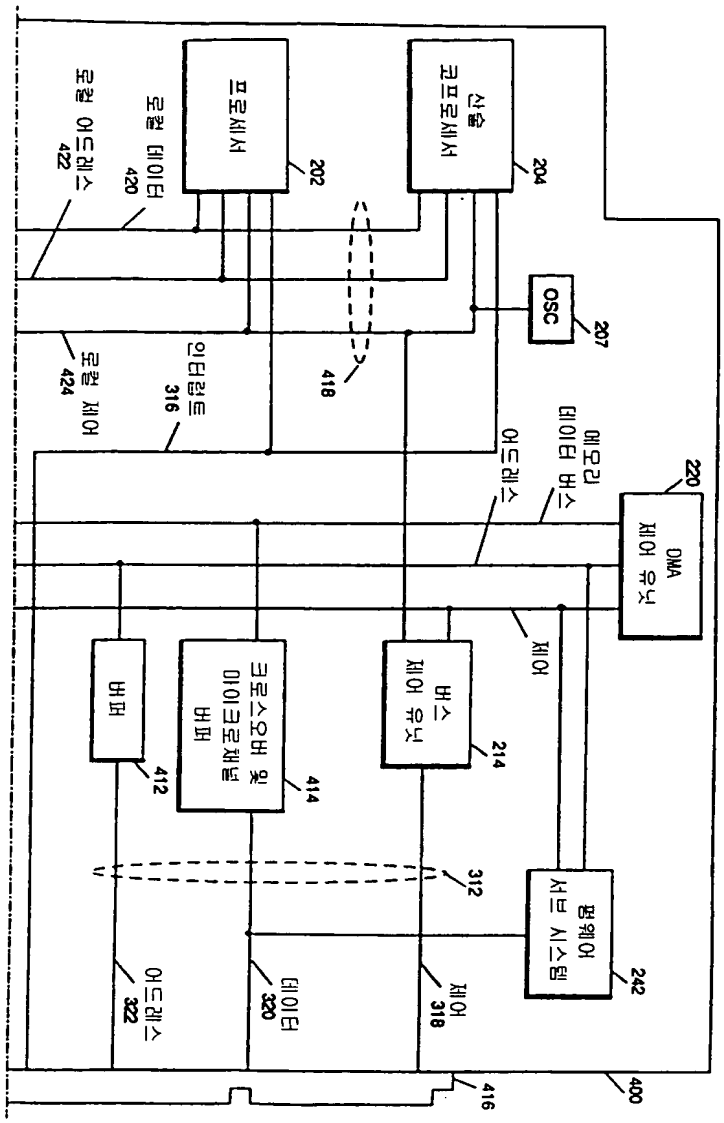
도면3c



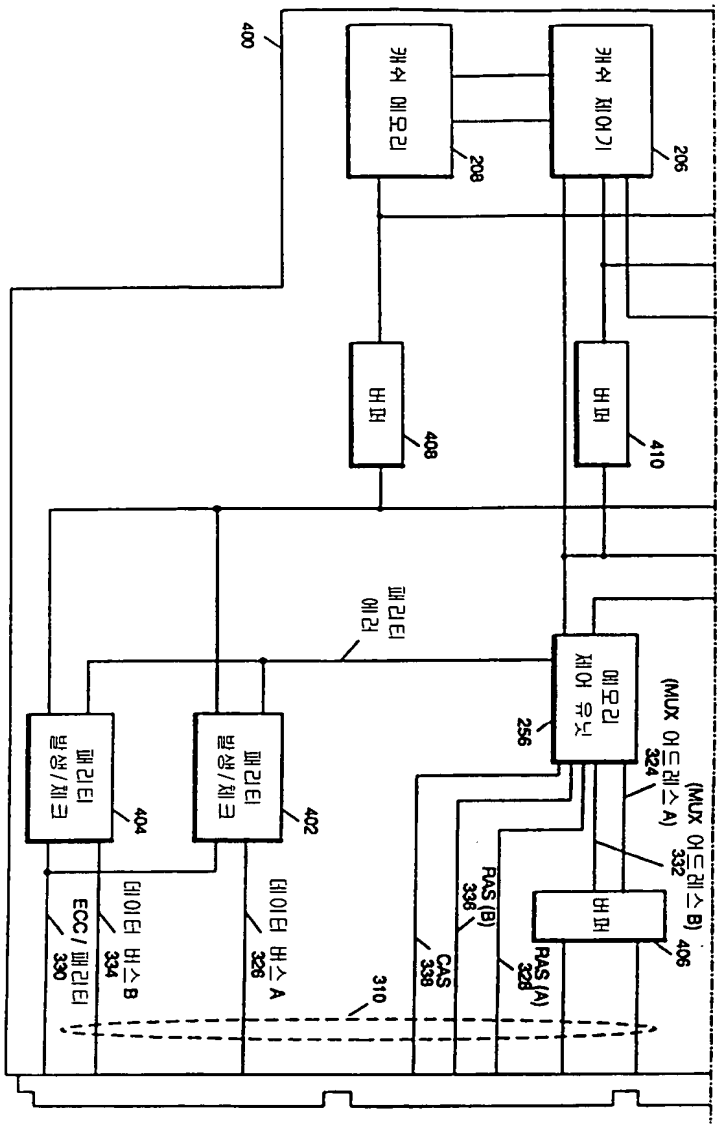
도면4

도 4a
도 4b

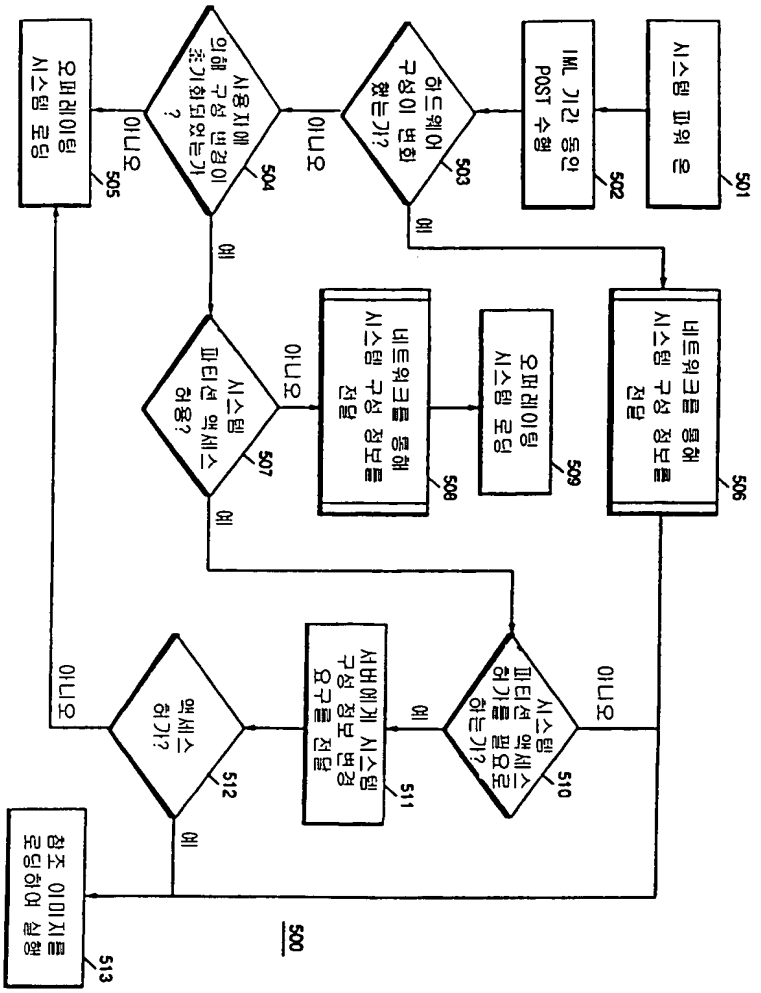
도면4a


프로세서
202
산술 코프로세서
204
로컬 데이터
420
로컬 어드레스
422
OSC
207
로컬 제어
424
인터럽트
316
418
에모리 데이터 버스
어드레스
220
DMA 제어 유닛
제어
버퍼
412
크로스오버 및 마이크로채널 버퍼
414
버스 제어 유닛
214
312
펌웨어 서브 시스템
242
어드레스
322
데이터
320
제어
318
416
400

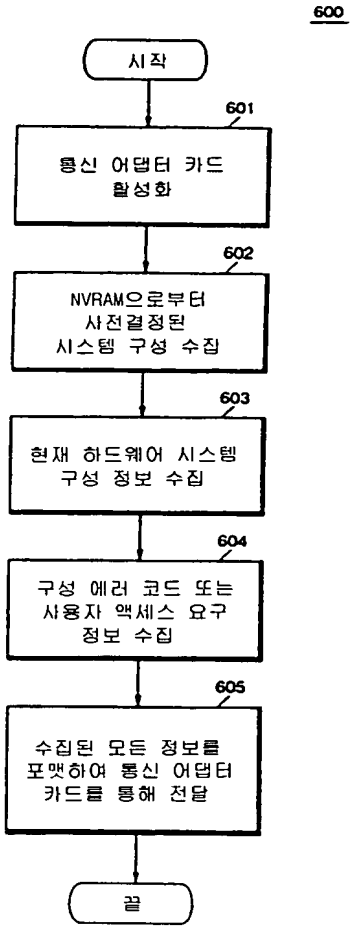
도면4b


400
캐쉬 메모리
208
캐쉬 제어기
206
버퍼
408
버퍼
410
패리티 에러
메모리 제어 유닛
256
(MUX 어드레스 A)
324
(MUX 어드레스 B)
332
버퍼
406
RAS (B)
336
RAS (A)
328
CAS
338
패리티 발생/체크
404
패리티 발생/체크
402
데이터 버스 B
334
ECC / 패리티
330
데이터 버스 A
326
310

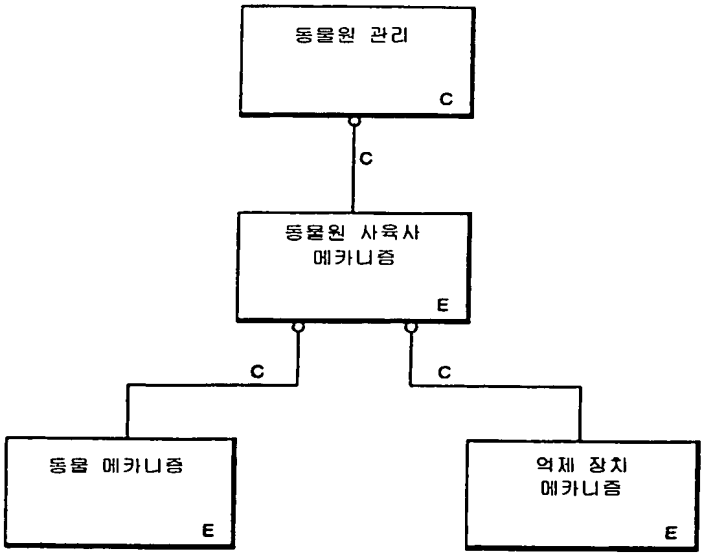
도면5


시스템 파워 온
501
IML 기간 동안 POST 수행
502
하드웨어 구성이 변화 했는가?
503
사용자에 의해 구성 변경이 초기화되었는가?
504
오퍼레이팅 시스템 로딩
505
네트워크를 통해 시스템 구성 정보를 전달
506
네트워크를 통해 시스템 구성 정보를 전달
508
오퍼레이팅 시스템 로딩
509
시스템 파티션 액세스 허용?
507
시스템 파티션 액세스 허가를 필요로 하는가?
510
서버에게 시스템 구성 정보 변경 요구를 전달
511
액세스 허가?
512
참조 이미지를 로딩하여 실행
513
예
아니오
500

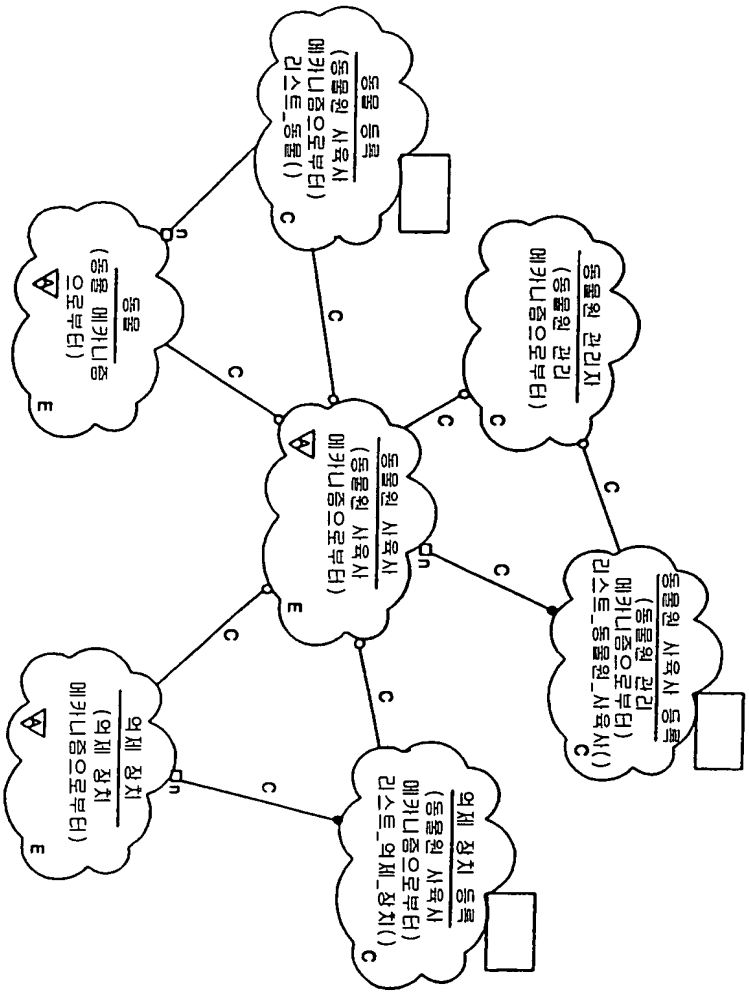
도면6


600
시작
601
통신 어댑터 카드
활성화
602
NVRAM으로부터
사전결정된
시스템 구성 수집
603
현재 하드웨어 시스템
구성 정보 수집
604
구성 에러 코드 또는
사용자 액세스 요구
정보 수집
605
수집된 모든 정보를
포맷하여 통신 어댑터
카드를 통해 전달
끝

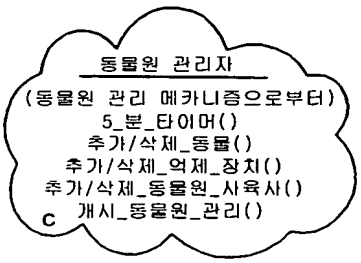
도면7


동물원 관리
C
C
동물원 사육사
메카니즘
E
C
C
동물 메카니즘
E
억제 장치
메카니즘
E

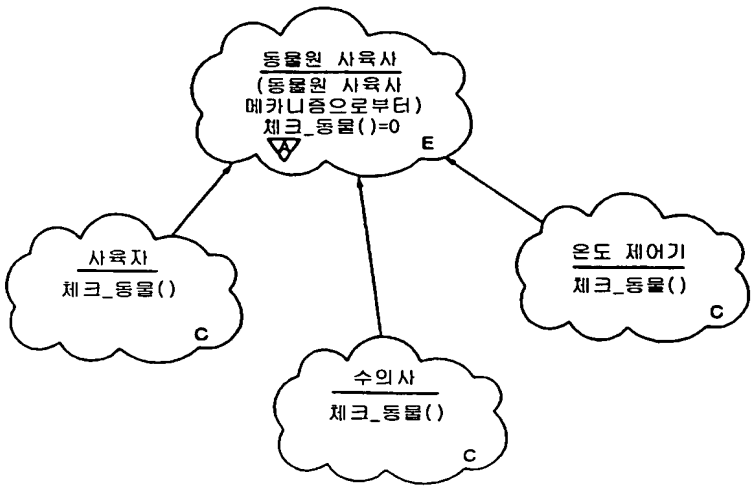
도면8


동물 등록
(동물원 사육사
메카니즘오로부터)
리스트_동물()
C
동물
(동물 메카니즘
오로부터)
E
동물원 관리자
(동물원 관리
메카니즘오로부터)
C
동물원 사육사
(동물원 사육사
메카니즘오로부터)
E
동물원 사육사 등록
(동물원 관리
메카니즘오로부터)
리스트_동물원_사육사()
C
억제 장치
(억제 장치
메카니즘오로부터)
E
억제 장치 등록
(동물원 사육사
메카니즘오로부터)
리스트_억제_장치()
C
C
C
C
C
C
C
C
C
C
n
n
n

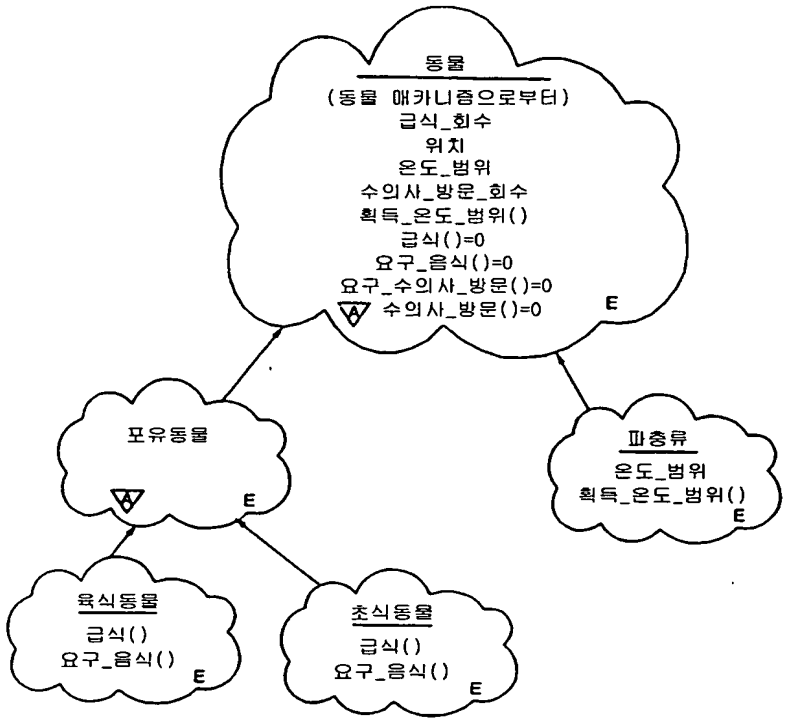
도면9


동물원 관리자
(동물원 관리 메카니즘으로부터)
5_분_타이머()
추가/삭제_동물()
추가/삭제_억제_장치()
추가/삭제_동물원_사육사()
개시_동물원_관리()
C

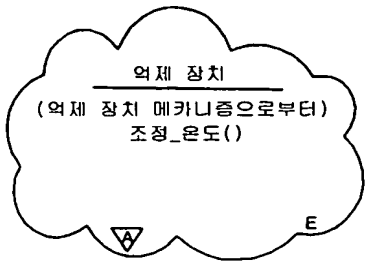
도면10



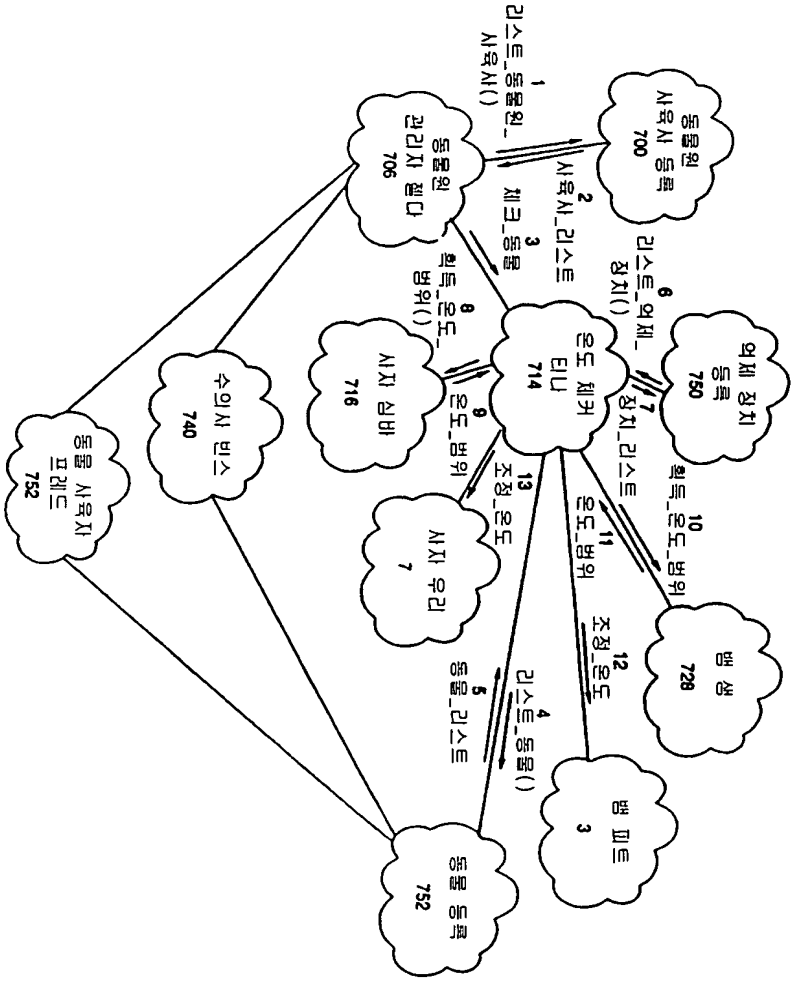
도면11



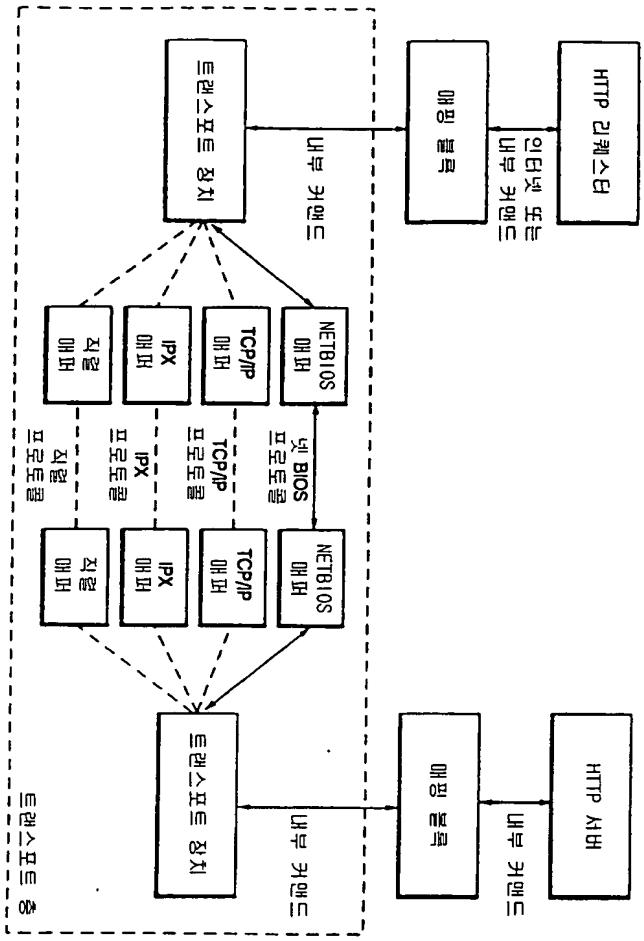
도면12



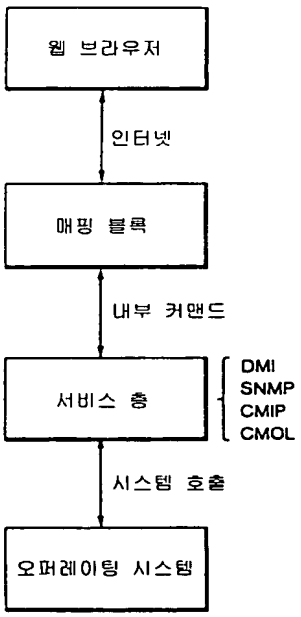
도면13


동물원 사육사 등록 700
동물원 관리자 젤다 706
온도 체커 티나 714
사자 심바 716
사자 우리 7
수의사 빈스 740
동물 사육자 프레드 752
동물 등록 752
억제 장치 등록 750
뱀 생 728
뱀 피트 3
1 리스트_동물원_사육사()
2 사육사_리스트
3 체크_동물
4 리스트_동물()
5 동물_리스트
6 리스트_억제_장치()
7 장치_리스트
8 획득_온도_범위()
9 온도_범위
10 획득_온도_범위
11 온도_범위
12 조정_온도
13 조정_온도

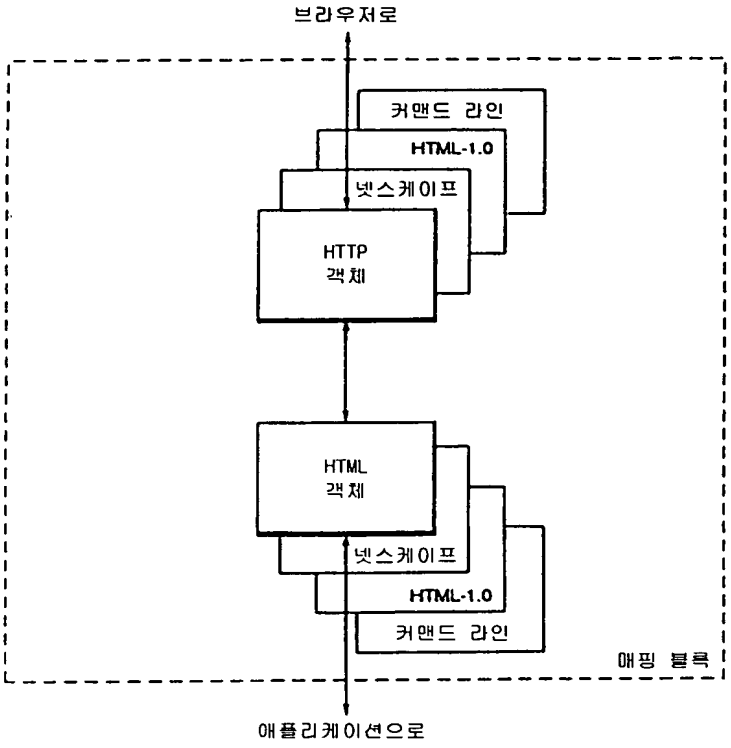
도면14a


트랜스포트 층
트랜스포트 장치
내부 커맨드
매핑 블록
내부 커맨드
HTTP 서버
트랜스포트 장치
내부 커맨드
매핑 블록
인터넷 또는 내부 커맨드
HTTP 리퀘스터
직렬 매퍼
IPX 매퍼
TCP/IP 매퍼
NETBIOS 매퍼
직렬 프로토콜
IPX 프로토콜
TCP/IP 프로토콜
넷 BIOS 프로토콜
직렬 매퍼
IPX 매퍼
TCP/IP 매퍼
NETBIOS 매퍼

도면14b


웹 브라우저
인터넷
매핑 블록
내부 커맨드
서비스 층
DMI
SNMP
CMIP
CMOL
시스템 호출
오퍼레이팅 시스템

도면15


브라우저로
커맨드 라인
HTML-1.0
넷스케이프
HTTP
객체
HTML
객체
넷스케이프
HTML-1.0
커맨드 라인
애핑 블록
애플리케이션으로